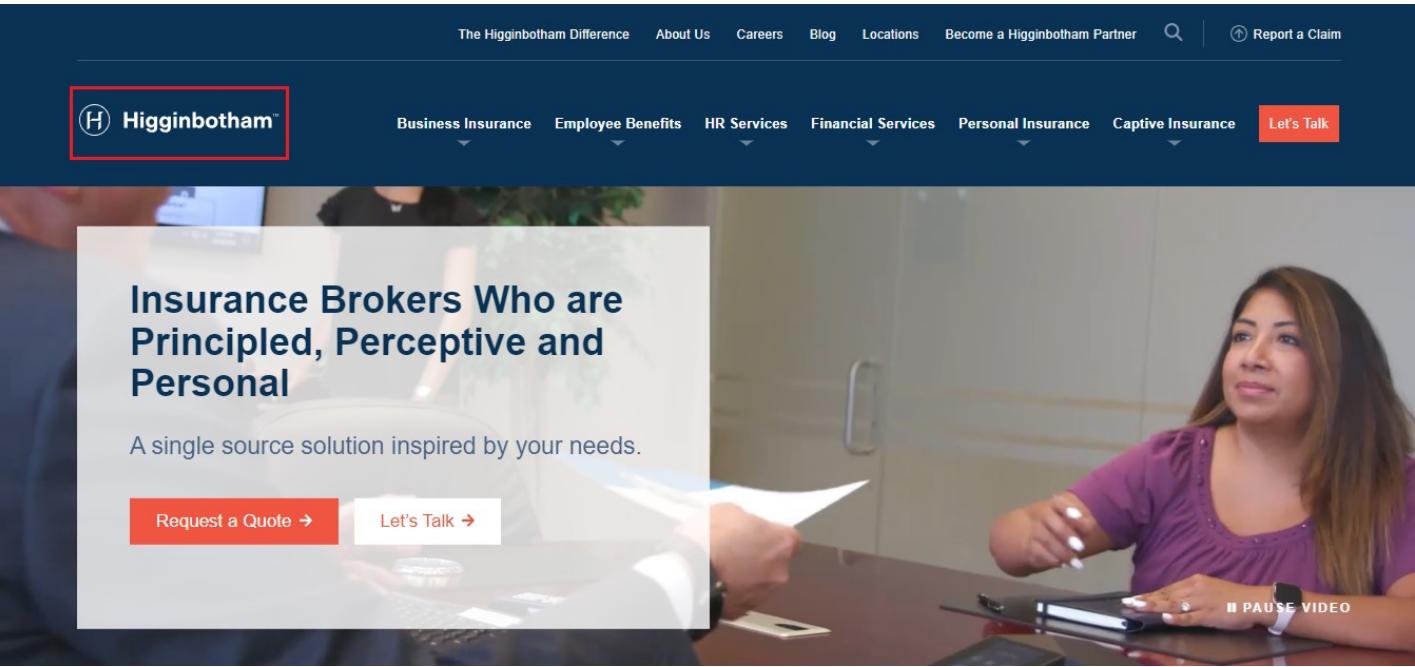


# Exhibit 2

|  |   |
|--|---|
| US7203844B1  | <p>Higginbotham Companies website <a href="https://www.higginbotham.com">higginbotham.com</a> ("The accused instrumentality")</p>   |
| <p>1. A method for a recursive security protocol for protecting digital content, comprising:</p> | <p>The accused instrumentality practices a method for a recursive security protocol (e.g., TLS 1.3 security protocol) for protecting digital content (e.g., digital certificate related to the accused instrumentality).</p> <p>The accused instrumentality utilizes TLS 1.3 security protocol (hereinafter "the standard") for communicating content such as digital certificate, application data, etc., with a client. The standard provides a two-level encryption security. It encrypts a plaintext with a first encryption technique and generates a ciphertext. Further, it encrypts the ciphertext with a second encryption technique i.e., recursive encryption security.</p>  <p><a href="https://www.higginbotham.com/">https://www.higginbotham.com/</a></p> |

The screenshot shows the Google Play Store interface. At the top, there is a search bar with the text "higginbotham". Below the search bar, there are three category tabs: "Apps & games" (selected), "Movies & TV", and "Books". On the right side of the header, there is a user profile icon and a help icon.

The main content area displays search results. The first result is highlighted with a red box and labeled "Higginbotham FSA Higginbotham Wex Health Mobile". It includes a brief description: "Save time and hassles with the Higginbotham FSA mobile app", a circular icon with a blue letter "H", a download count of "1K+", and a rating of "Everyone". A large "Install" button is present. To the right of this result, there are three screenshots of the app's interface:

- The first screenshot shows the "My Accounts" screen with sections for FSA, HSA, and DependentCare, each showing a balance of \$2,400.00.
- The second screenshot shows the "Account" screen for the FSA account, displaying details like Available Balance (\$2,400.00), First Service Date (12/01/2018), and Plan Ending Date (03/31/2021).
- The third screenshot shows the "Dashboard" screen with a "CREATE NEW EXPENSE" button and a list of expenses categorized as Pharmacy, Doctor, and Medical.

At the bottom of the page, there is a purple horizontal bar containing the URL <https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US>. To the right of the URL, there are links for "Privacy - Terms" and a small circular icon with a blue letter "G".

The screenshot shows a web browser displaying the Higginbotham website. The header features a large blue 'H' logo inside a circle, followed by the brand name "Higginbotham™" in a bold, dark blue sans-serif font. A horizontal menu bar is visible above the main content area. On the left side of the menu bar, there is a blue button labeled "Welcome". To the right of the menu bar, the date "Monday, September 2, 2024" is displayed. Below the menu bar, a blue navigation bar contains the text "You are here: Welcome". The main content area is titled "Account Login" and contains a form with a red border. The form includes a label "User name:" and a text input field. Below the input field is a blue "Login" button. Underneath the button, there is a link "Retrieve login information". At the bottom of the page, a URL is displayed: <https://higginbotham.secureclient.net>Welcome/tabid/366042/Default.aspx?returnurl=%2f>.

|  |   |
|--|---|
|  | <p>Security overview</p> <p>  </p> <hr/> <p>This page is secure (valid HTTPS).</p> <p> Certificate - valid and trusted<br/>The connection to this site is using a valid, trusted server certificate issued by GlobalSign Atlas R3 DV TLS CA 2024 Q3.<br/><a href="#">View certificate</a></p> <p> Connection - secure connection settings<br/>The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_128_GCM.</p> <p> Resources - all served securely<br/>All resources on this page are served securely.</p> <p><a href="https://higginbotham.secureclient.net&gt;Welcome/tabid/366042/Default.aspx?returnurl=%2f">https://higginbotham.secureclient.net&gt;Welcome/tabid/366042/Default.aspx?returnurl=%2f</a></p> |
|--|---|

The Transport Layer Security (TLS) Protocol Version 1.3

## Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

As shown below, the accused instrumentality utilizes a two-level algorithm security. It utilizes the SHA256RSA encryption algorithm as a first encryption algorithm i.e., signature encryption algorithm and the TLS\_AES\_256\_GCM\_SHA384 encryption algorithm as a second encryption algorithm i.e., AEAD encryption algorithm.

| 8  | 200 | HTTP  | Tunnel to www.higginbotham.com:443                   | 3,612  |  |
|----|-----|-------|--|--------|--|
| 9  | 200 | HTTPS | www.higginbotham.com /                               | 17,353 |  |
| 10 | 200 | HTTPS | www.googleapis.com /oauth2/v4/token                  | 399    |  |
| 11 | 200 | HTTPS | optimizationguide-p... /v1:Get-Links?                | 428    |  |
| 12 | 200 | HTTP  | Tunnel to safebrowsing.google.com...                 | 10,052 |  |
| 13 | 200 | HTTPS | safebrowsing.google.com /safebrowsing/clientrepor... | 32     |  |
| 14 | 200 | HTTPS | www.higginbotham.com /wp-includes/css/dist/bloc...   | 14,716 |  |
| 15 | 200 | HTTPS | www.higginbotham.com /wp-content/cache/min/1/...     | 35,238 |  |
| 16 | 200 | HTTP  | Tunnel to cdnjs.cloudflare.com:443                   | 2,748  |  |
| 17 | 200 | HTTP  | Tunnel to cdn.callrail.com:443                       | 4,277  |  |
| 18 | 200 | HTTPS | www.higginbotham.com /wp-content/cache/min/1/...     | 6,980  |  |
| 19 | 200 | HTTPS | www.higginbotham.com /wp-content/cache/min/1/...     | 23,260 |  |
| 20 | 200 | HTTPS | www.higginbotham.com /wp-content/cache/min/1/...     | 23,266 |  |
| 21 | 200 | HTTPS | www.higginbotham.com /wp-content/themes/orbit...     | 26,907 |  |
| 22 | 200 | HTTPS | www.higginbotham.com /wp-content/cache/min/1/...     | 1,436  |  |
| 23 | 200 | HTTPS | www.higginbotham.com /wp-includes/js/jquery/jqu...   | 31,396 |  |

*Source: Fiddler Capture*

SignedCertTimestamp(RFC6962) empty  
ALPN h2,11  
signature alg\$ ecdsa secp256r1 sha256 rsa\_pss\_rsae sha256,rsa\_pkcs1\_sha256,ecdsa\_secp384r1 sha384,rsa\_pss\_rsae sha384,rsa\_pkcs1\_sha384,rsa\_pss\_rsae sha512,rsa\_pkcs1\_sha512  
0x001b 02 00 02  
supported\_groups grease\_0x5a5a, unknown [0x6399], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]  
key\_share 04 ED 5A 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 F2 32 5D 09 69 67 98 9F 88 8B D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85  
7A 95 44 34 07 19 C8 33 CF EC 9C 45 46 08 B4 C9 1C 22 1E 3C 97 81 A3 C5 73 B2 80 E1 C9 53 CA 9D 44 B9 05 5B 9D A2 E4 2B 5F B5 82 84 67 9C C6 65 5F 82 3E B2 F0 B5 23 AF 0F 86 11 A4 1C 00 5A 46 D9 57 AF 31 C6 C2 F4 00 38 F7  
EC 97 6A 21 78 C4 8A BA 84 85 C9 85 67 D7 T9 OC AB 79 1D 83 3A 3B 82 60 D6 A3 C2 01 87 A7 F7 CC B8 A0 E5 1F 55 88 34 65 7F 2B CA 06 9C 3B 17 9D A6 11 B2 39 9F 34 00 6C BD C0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E F7  
C0 61 1D A0 54 55 73 A1 99 23 8C 39 BA A0 C6 73 TA 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A B5 2C 21 80 CA 1F E2 AC B4 73 A5 FD EB 7B 6D B5 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F E0 75 BA 39 AA OB 66 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F3  
C1 23 C1 5E 05 08 13 61 A7 C1 26 7B D7 FB 9F E9 09 14 5A E3 B6 5A 5B 9D D2 D5 47 0D 6B 7A 4A 71 6D 0E 01 37 F8 ED A3 E1 58 6B D6 D2 C0 26 53 B8 88 B0 09 DF 5A 20 22 E9 A3 D3 DD 00 9E 98 F0 4C AD 11 99 AD C0 48 DA D1 37 3B 2A AF 72 43 91 BD 66 23 56 C7 5A E0 5E 74 33 BF 3A 86 40 07 F7 T9 95 B1 38 70 23 05 30 1B 5E A5 C7 07 13 84 84 A9 90 83 66 B5 11 A9 A6 1 2 3C 15 DC EC 94 3E 55 93 83 64 CO BF 79 2F C3 22 6C B1 C4 AB 5B 12  
83 A0 BA 02 8F 26 58 2F 47 7B 2B 10 49 21 80 15 54 13 60 DA 90 E3 A0 82 98 C8 86 63 7A A8 AA E7 08 2D 4B 10 BA 70 A0 3A 93 D1 75 07 59 1C 7D 8C 90 55 26 AB 90 91 C9 72 10 B8 99 3C E8 A3 A7 F0 73 D4 FA 81 9F 99 BF 70 7B A1  
08 5B 88 AC 2C CD 1F FB BB 73 23 89 CO F9 82 BB A8 65 B3 E4 6C 58 59 20 85 53 12 EB 49 EB A3 13 B7 8A F5 1A 31 70 86 88 88 96 96 7F 23 9C 37 19 0C CO 03 CB 14 8F 78 7A F5 03 04 04 51 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 44 AB  
CA 6B 3C A4 85 1D 6E 70 74 86 C7 9D B9 88 4D DD 4B 39 6D AB 00 9C 52 17 C3 C3 AD F6 10 A0 DE 0A B5 86 C7 49 5D 94 3A C5 39 95 02 BC 1E A2 73 6A 2D 16 1C 54 31 68 1D A6 C9 B5 C3 A0 12 3B 1C 72 10 60 3E 3B 23 6A 68 92 DF CC 86 91 E7 98  
7B 22 01 7B 34 03 07 61 05 82 93 54 35 30 23 74 A3 07 84 0D 91 77 7A 13 45 4D 4C 82 E6 68 94 3D 20 34 22 4F 29 27 32 45 08 33 92 FC 4D A9 13 3A 44 8A 23 98 EC 22 E9 01 24 26 9B 8D CO 0B 45 B9 48 83 AD BA E4 99 3A 23 23 32 6C 68  
31 26 FC 09 97 C2 2E 7D 39 89 BA A1 77 69 3C 94 86 FA C3 3B C9 E4 5C 75 BC 59 88 60 14 21 26 83 FF 72 2C 0D F7 E4 88 80 33 67 FD C2 C0 1A 5B 69 43 61 58 43 94 FA 9C 67 00 03 7A 8C 6E C4 4E 2C 49 E0 82 8C DA BB AF DA 6A 6D B5 08  
9D CA A9 A5 5A E2 91 55 3C 03 C0 B0 D2 AD C6 E1 A9 8F C4 54 15 88 3C A3 21 21 28 CD AC 99 09 A2 37 B4 46 53 2F AF E4 AF E8 F2 55 CC FC 4F 4F 78 A4 FC 28 64 10 17 C6 61 71 2F 90 25 26 CO 05 38 2A C6 44 99 9E F8 D6 3B 1C  
48 A0 CE DC B0 B6 1C 85 B3 75 4F B4 5A 14 3F 09 C0 ED 65 E7 18 12 36 34 C1 B5 A3 1B E4 17 72 F1 2F 85 20 88 21 68 B1 AC 3C E4 44 2E 53 6C CO 0A 82 32 07 76 C5 23 8C CO 04 07 00 CO B0 C2 44 48 D3 B7 13 00 11 C5 88 73 26 F2 EA 96 15 55  
CC 3E EB 0F 80 28 93 D4 49 B4 1D 52 18 44 88 86 F5 BC AD B8 60 BF F2 BC 1E 43 34 23 C8 57 4B 65 2F E6 29 97 C8 39 3D B7 E7 0F 2D C2 AF 53 DA AA 56 C1 7D 4B 93 54 D9 67 C5 D9 5F 1E 7B 86 49 57 45 85 58 56 84 32 42 B5 39 60 62 40 C2 96  
B1 24 9A DA 84 BD 8F DA 98 9C SC 05 82 17 19 12 FD FA 32 DB 69 C3 B3 T8 7B 84 15 70 C9 69 73 T8 A6 D7 A1 21 4C F0 63 06 42 37 FD F3 A3 B5 D0 85 EF D6 02 FD F1 6D 4A 3C 55 AB BA 8A 0F 45 A6 5C C8 53 19 F6 A1 75 06 90 A0 26 2D A3 90  
H9 18 53 C2 E9 0F 05 83 3D 06 49 73 33 8D 4B 8E 72 29 2F FB 51 38 42 5F 76 C4 AE 54 45 E6 91 8D 28 45 F0 9A 1E 65 C0 3A D7 99 F5 AE F1 9D E9 45 D3 A3 CD DB 7A 74 E0 3D ED 7F 6D 00 1D 00 20 FA D2 73 2A TD BB 08 13 72 9B 38 A0 89 F4  
7A 36 23 01 54 D1 F6 FB 09 92 92 32 D7 CC C6 34 38 24  
extended\_master\_secret empty  
ec\_point\_format uncompressed [0x0]  
status\_request OCSP - Implicit Responder  
psk\_key\_exchange\_modes 01 01  
renegotiation\_info 00  
0xfe0d 00 00 01 00 01 8B 00 20 08 F0 C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15  
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 B5 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E  
2D 8A 8B F9 05 D4 63 1E F5 35 82 4F 90 88 5C AB 89 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 B6 F1 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C  
E4 C8 4A D5 F3 3C 6D 8F 6A DF C0 53 9F 27 E4 C6 05 53 5F 83 38 B5 A  
server\_name www.higginbotham.com  
server\_port 443  
server\_aliases

*Source: Fiddler Capture*

*Source: Fiddler Capture*

[Thumbprint]  
129157F7A5852AD23DF17FE2934E6396DB6B0251

[Signature Algorithm]  
sha256RSA(1.2.840.113549.1.1.11)

First decryption algorithm

[Public Key]

Algorithm: RSA

Length: 2048

Key Blob: 30 82 01 0a 02 82 01 01 00 ca b8 66 5f 8a 37 94 14 e1 2f e3 49 cf d7 ad cb 20 8c 54 93 6b af 66 56 02 11 26 db 20 13 00 a0 ce fe ae e9 ac 8d 68 8f a0 56 fd ff b4 14 b9 c5 54 ff 55 4a 95 76 b9 db 3c 77 5e 9a b1 dd 30 74 36 77 cb 92 12 df 17 62 6d 2a aa 74 3c f6 68 c8 34 8d eb 69 97 9e f2 43 02 fa d7 76 99 c0 bc b3 56 2d 17 ca 39 c8 00 5d 91 c8 1d 9b c9 70 c8 c1 b0 da 40 23 3e 9a fe a9 ed e4 60 ce 15 b0 af 43 a3 07 a5 81 25 dd c3 92 e7 72 9e 27 df bd 01 05 2a b4 60 b8 83 6b fa bc db 97 51 47 a8 2e d0 52 f7 c2 0a 1c 02 96 d2 c7 fd 04 a5 7e 78 04 ee 59 ed 33 9a e4 cc 42 bb 87 36 95 ac 4b 9b 6b 4f 96 4c ed 20 f7 bd 71 4b bc d8 91 81 3c be 56 c8 05 a6 e1 f7 66 85 88 46 91 96 a3 e4 e8 bb 45 f5 b8 78 25 4b 02 cb 23 e6 54 94 ed 94 f6 87 3d 29 55 31 eb 59 e9 a1 54 4c 50 b0 74 2c a7 b6 1e f7 23 02 03 01 00 01  
Parameters: 05 00

[Extensions]

\* Key Usage(2.5.29.15):

Digital Signature, Key Encipherment (a0)

*Source: Fiddler Capture*

| Headers  | Text View  | Syntax View | Web Forms | Hex View | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm   |
|----------|--|-------------|-----------|----------|------|---------|-----|------|-----|---|
| 00000019 | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 |             |           |          |      |         |     |      |     | com:443 HTTP/1.1..Host: www.higginbotham.com:443..Connection: keep-alive..User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36...A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below...Secure |
| 00000032 | 77 77 2E 68 69 67 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A |             |           |          |      |         |     |      |     | Protocol: TLS 1.3.Cipher Suite: TLS_AES_256_GCM_SHA384..Record Layer Version: 3.3 (TLS/1.2).Random:   |
| 0000004B | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55 |             |           |          |      |         |     |      |     | F6 9D 1E 05 3D 58 53 50 C5 38 CB 68 E9 B1 71 BE 0   |
| 00000064 | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 |             |           |          |      |         |     |      |     | 2 77 A7 FA AB 3F CC 1D 97 1B 4C AF AB 7A CD 69."Time": "21-09-1972 08:33:18. SessionID: 5E B3 4B 70 12 4D 2C CB 6A 5B 9A 63 89  |
| 0000007D | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36 |             |           |          |      |         |     |      |     |   |
| 00000096 | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48 |             |           |          |      |         |     |      |     |   |
| 000000AF | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31 |             |           |          |      |         |     |      |     |   |
| 000000CB | 32 36 2E 30 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D |             |           |          |      |         |     |      |     |   |
| 000000E1 | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E |             |           |          |      |         |     |      |     |   |
| 000000FA | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E |             |           |          |      |         |     |      |     |   |
| 00000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20 |             |           |          |      |         |     |      |     |   |
| 0000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65 |             |           |          |      |         |     |      |     |   |
| 00000145 | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72 |             |           |          |      |         |     |      |     |   |
| 0000015E | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53 |             |           |          |      |         |     |      |     |   |
| 00000177 | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69 |             |           |          |      |         |     |      |     |   |
| 00000190 | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A |             |           |          |      |         |     |      |     |   |
| 000001A9 | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20 |             |           |          |      |         |     |      |     |   |
| 000001C2 | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30 |             |           |          |      |         |     |      |     |   |
| 000001DB | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37 |             |           |          |      |         |     |      |     |   |
| 000001F4 | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69 |             |           |          |      |         |     |      |     |   |
| 0000020D | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A |             |           |          |      |         |     |      |     |   |
| 00000226 | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32 |             |           |          |      |         |     |      |     |   |
| 0000023F | 20 34 44 20 32 43 20 43 42 20 36 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |             |           |          |      |         |     |      |     |   |

*Source: Fiddler Capture*

|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 39 2C 7B 89 AA BB C5 AF E |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | 0C 2A 73 3E B4 85 7A 95   |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |

*Source: Fiddler Capture*

Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.

Secure Protocol: TLS 1.3  
Cipher Suite: TLS\_AES\_256\_GCM\_SHA384

-- Server Certificate -----  
[Version]  
V3

[Subject]  
CN=higginbotham.com  
Simple Name: higginbotham.com  
DNS Name: higginbotham.com

[Issuer]  
CN=GTS CA 1P5, O=Google Trust Services LLC, C=US

*Source: Fiddler Capture*

The standard defines four record message types, including a handshake message type.

The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## 2. Protocol Overview

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

All handshake messages after the ServerHello are now encrypted. The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

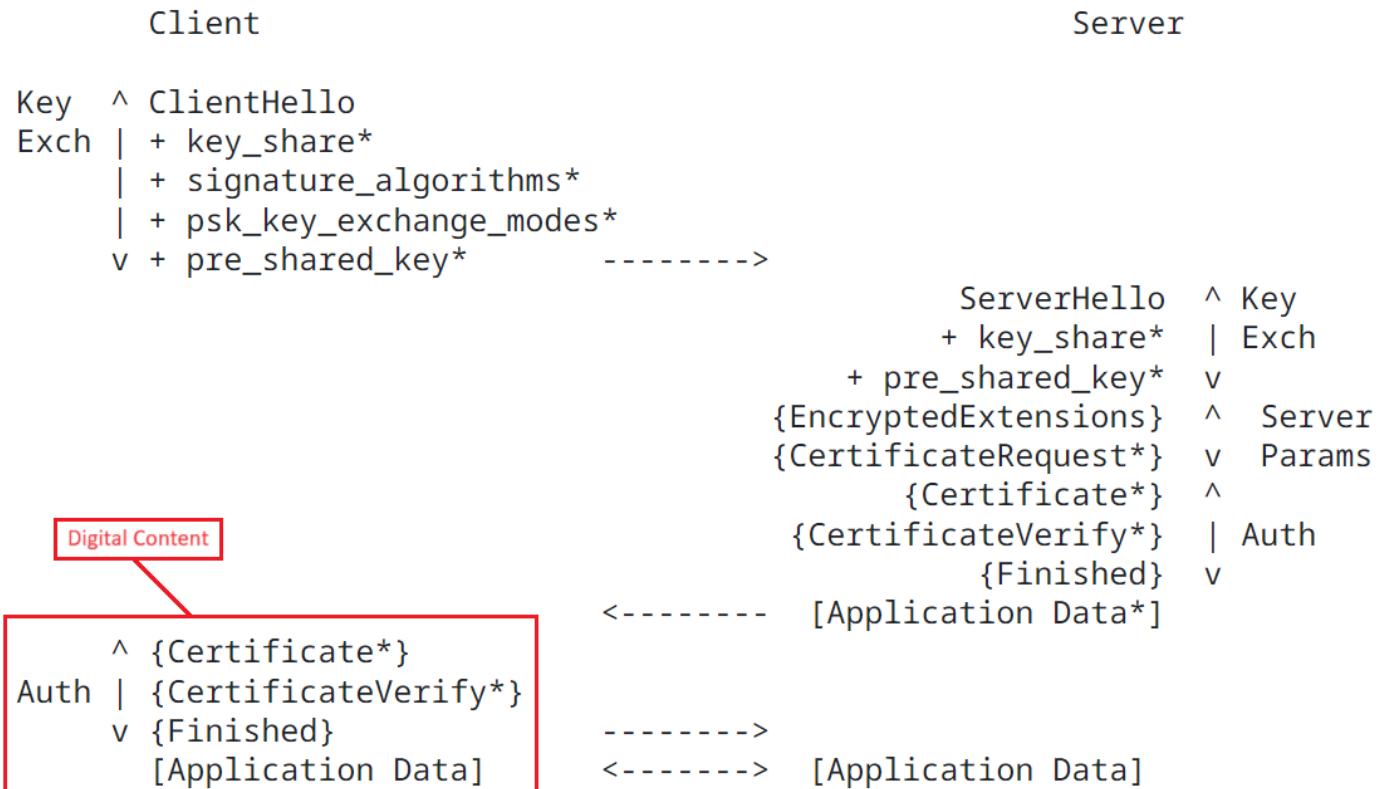
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## Introduction

The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream. Specifically, the secure channel should provide the following properties:

- Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated.  
Authentication can happen via asymmetric cryptography (e.g., RSA [[RSA](#)], the Elliptic Curve Digital Signature Algorithm (ECDSA) [[ECDSA](#)], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [[RFC8032](#)]) or a symmetric pre-shared key (PSK).
- Confidentiality: Data sent over the channel after establishment is only visible to the endpoints. TLS does not hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.
- Integrity: Data sent over the channel after establishment cannot be modified by attackers without detection.

First encryption

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [\[RFC5116\]](#), [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   | <p>This specification defines the following cipher suites for use with TLS 1.3.</p> <table border="1"><thead><tr><th>Description</th><th>Value</th></tr></thead><tbody><tr><td>TLS_AES_128_GCM_SHA256</td><td>{0x13,0x01}</td></tr><tr><td>TLS_AES_256_GCM_SHA384</td><td>{0x13,0x02}</td></tr><tr><td>TLS_CHACHA20_POLY1305_SHA256</td><td>{0x13,0x03}</td></tr><tr><td>TLS_AES_128_CCM_SHA256</td><td>{0x13,0x04}</td></tr><tr><td>TLS_AES_128_CCM_8_SHA256</td><td>{0x13,0x05}</td></tr></tbody></table> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> | Description | Value | TLS_AES_128_GCM_SHA256 | {0x13,0x01} | TLS_AES_256_GCM_SHA384 | {0x13,0x02} | TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} | TLS_AES_128_CCM_SHA256 | {0x13,0x04} | TLS_AES_128_CCM_8_SHA256 | {0x13,0x05} |
|---|--|-------------|-------|------------------------|-------------|------------------------|-------------|------------------------------|-------------|------------------------|-------------|--------------------------|-------------|
| Description   | Value  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_128_GCM_SHA256                                    | {0x13,0x01}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_256_GCM_SHA384                                    | {0x13,0x02}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_CHACHA20_POLY1305_SHA256                              | {0x13,0x03}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_128_CCM_SHA256                                    | {0x13,0x04}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_128_CCM_8_SHA256                                  | {0x13,0x05}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| encrypting a bitstream with a first encryption algorithm; | <p>The standard practices encrypting a bitstream (e.g., bitstream of digital certificate) with a first encryption algorithm (e.g., signature encryption algorithm i.e., SHA256RSA encryption algorithm).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA encryption algorithm) and generates a ciphertext.</p>   |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

As shown below, the accused instrumentality discloses the signature encryption algorithm.

|    |     |       | Tunnel to              | www.higginbotham.com:443             |        |
|----|-----|-------|------------------------|--------------------------------------|--------|
| 8  | 200 | HTTP  |                        |                                      | 3,612  |
| 9  | 200 | HTTPS | www.higginbotham....   | /                                    | 17,353 |
| 10 | 200 | HTTPS | www.googleapis.com     | /oauth2/v4/token                     | 399    |
| 11 | 200 | HTTPS | optimizationguide.p... | /v1:Gethints?                        | 428    |
| 12 | 200 | HTTP  |                        | Tunnel to safebrowsing.google.com... | 10,052 |
| 13 | 200 | HTTPS | safebrowsing.googl...  | /safebrowsing/clientrepor...         | 32     |
| 14 | 200 | HTTPS | www.higginbotham....   | /wp-includes/css/dist/bloc...        | 14,716 |
| 15 | 200 | HTTPS | www.higginbotham....   | /wp-content/cache/min/l/...          | 35,238 |
| 16 | 200 | HTTP  |                        | Tunnel to cdnjs.cloudflare.com:443   | 2,748  |
| 17 | 200 | HTTP  |                        | Tunnel to cdn.callall.com:443        | 4,277  |
| 18 | 200 | HTTPS | www.higginbotham....   | /wp-content/cache/min/l/...          | 6,980  |
| 19 | 200 | HTTPS | www.higginbotham....   | /wp-content/cache/min/l/...          | 23,260 |
| 20 | 200 | HTTPS | www.higginbotham....   | /wp-content/cache/min/l/...          | 23,266 |
| 21 | 200 | HTTPS | www.higginbotham....   | /wp-content/themes/orbit...          | 26,907 |
| 22 | 200 | HTTPS | www.higginbotham....   | /wp-content/cache/min/l/...          | 1,436  |
| 23 | 200 | HTTPS | www.higginbotham....   | /wp-includes/js/jquery/jqu...        | 31,396 |

Transformer Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML

Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.

Secure Protocol: TLS 1.3  
Cipher Suite: TLS\_AES\_256\_GCM\_SHA384

== Server Certificate ======  
[Version]  
V3  
  
[Subject]  
CN=higginbotham.com  
Simple Name: higginbotham.com  
DNS Name: higginbotham.com  
  
[Issuer]  
CN=GTS CA 1P5, O=Google Trust Services LLC, C=US  
Simple Name: GTS CA 1P5  
DNS Name: GTS CA 1P5

*Source: Fiddler Capture*

|  |  |
|--|--|
|  | <p>SignedCertTimestamp (RFC6962) empty</p> <p>Digital certificate</p> <p>ALPN h2, http/1</p> <p>signature algs ecdsa secp256r1 sha256, rsa pss rsae sha256, rsa pkcs1 sha256, ecdsa secp384r1 sha384, rsa_pss_rsae_sha384, rsa_pkcs1_sha384, rsa_pss_rsae_sha512, rsa_pkcs1_sha512</p> <p>0x001b 02 00 02</p> <p>supported_groups grease [0x5a5a], unknown [0x6399], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]</p> <p>key_share 04 ED 5A 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 F2 32 5D 09 69 67 98 9F 8B 88 D0 A6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85</p> <p>7A 95 44 34 07 19 C3 33 3F EC 95 45 06 84 C9 21 C2 E1 3C 9F 87 71 A3 5B C3 72 B0 E1 C9 D5 CA 9D C4 69 80 05 5B 9D E2 A4 28 56 F5 FB 82 84 67 9C C6 65 5F 82 E2 F0 BB F5 3B AF E0 86 11 A4 1C 00 5A 46 89 D9 57 AF 31 C6 C2 F4 F0 38 F1</p> <p>EC 97 9A 21 78 C4 A8 AA B4 85 C9 C5 87 D7 70 CA EB AB 79 1D 83 3A B2 3B 60 D6 A3 C2 01 87 A7 F7 CC B8 A0 E5 1E 1F 55 88 34 65 2F BA CA 06 9C 3B 17 9D A6 11 B2 39 9F 34 00 6C CB D0 53 3E 57 16 06 72 B7 D8 42 36 EA 03 9B DB 8E 2F</p> <p>C0 61 D0 A4 55 73 A1 99 23 8C 39 BA A0 C6 73 TA 95 49 8A 6B A1 69 74 99 49 55 5B BC OA B5 2C 21 80 CA 1F E2 AC B4 73 A5 FD EB 78 6D B5 93 D6 10 37 A1 E3 C8 18 C6 49 D3 71 0F E0 75 39 AA 0B 86 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F5</p> <p>23 C1 E6 5E E8 03 13 61 F0 A8 47 A3 00 28 E6 A7 C1 C2 6B D7 F9 EB 09 14 5A E3 B6 5A 5B 98 D2 D5 47 0D 6B 78 A1 71 6D 0E 01 37 F8 E0 A3 1E 58 6B D6 D2 C0 26 53 B8 98 80 09 DF 5A 20 22 E3 DD 09 E6 98 F0 4C A0</p> <p>11 99 AD C0 48 DA D1 37 3B 2A AF 72 43 91 BD 6E 23 56 C7 5A EO 5E 74 33 BF 3A 86 40 07 F7 7D 95 B1 38 70 23 05 30 1B 5E A5 C7 07 13 84 A8 A9 80 B3 66 B5 11 A9 A1 61 72 3C 15 DC EC 94 3E 55 93 83 64 CO BF 79 2F C3 22 6C B1 C4 AB 5B 12</p> <p>83 A0 B2 86 28 9F 47 47 7B 20 81 5F 13 53 0B 8E A0 82 B8 63 TA 7A 90 A3 91 D1 75 07 59 1C 1D C8 59 55 26 B4 A9 90 91 C7 92 10 EB 82 99 EB A3 A7 F0 73 D4 FA 81 9F 99 BF 87 B7 A1</p> <p>08 BB 58 A8 CC 2D C1 FB BB 77 23 89 CF B9 92 BB A8 65 B3 E4 6C 68 59 20 88 B5 53 12 EB 49 A3 13 B3 78 A5 F1 3A 70 86 88 96 96 7F 23 9C 37 19 0C C0 3C 08 14 4F 67 78 4A F5 03 04 61 CB E4 AD 37 D0 65 57 26 89 24 DC 8E 15 A4 AB</p> <p>CA 6B 3C 4A 85 1D 6E 7C 9D B9 88 4D DD 4B 39 6D AD 00 9C 52 17 C3 C4 AD 10 A0 DE 04 B5 86 C7 49 6D 94 3A C5 39 95 02 BC 1E A2 73 6A 2D 16 1C 54 31 68 1D A6 C9 B5 C3 A0 12 3B 1C 72 10 60 3E 3B 23 6A 68 92 DF CC 86 91 E7 98</p> <p>2B 72 01 7B 34 07 61 05 82 53 94 6E 2C CC 30 74 A3 A7 08 4D 91 77 7A 14 5C 4D 4F E2 68 66 94 AD 20 34 22 4F 29 2B 29 37 42 25 03 9B F4 CD A9 19 3A 44 8A 23 98 EC 22 E9 01 24 6D 29 B8 DC 08 0B 45 88 43 BA E4 99 3A 32 23 32 EC 66</p> <p>31 26 FC 06 97 C2 2E TD 39 B9 AA 17 69 3C 94 66 FA C3 3B 9B E4 75 BC 59 8B 60 14 A7 26 83 FF 27 2C 0D F7 4F E9 88 30 63 67 FD F3 C2 C0 1A 5B 69 2F 61 48 53 94 9A FC 67 00 03 7A 8C 8E CB 4E 4C 2C 49 0B 82 DC DA 88 AF DA 6A 6D B5 08</p> <p>9D CA 5A E2 91 55 26 E0 50 5A 3C A9 21 62 83 FF 27 2C 0D F7 4F E9 88 30 63 67 FD F3 C2 C0 1A 5B 69 2F 61 48 53 94 9A FC 67 00 03 7A 8C 8E CB 4E 4C 2C 49 0B 82 DC DA 88 AF DA 6A 6D B5 08</p> <p>48 C0 AE DC B0 86 1C 85 B3 75 4F 84 5A 14 3F C0 90 CD E6 B7 18 12 6E 34 52 B1 C3 A1 B4 17 72 1F F8 25 20 88 21 6B B1 AC 3C E4 4A 2E 53 6C CC 0A 82 32 07 F5 C5 23 8C C0 40 07 00 C0 B0 C2 4A 48 D3 B7 13 00 11 C5 83 73 26 F2 EA 96 15 55</p> <p>CC 3E EB 08 28 93 D4 49 B3 1D 52 18 88 65 F5 BC A8 60 B6 F2 B2 C8 37 B4 46 53 2F AF E4 AF E8 22 55 CC FC 4F 78 A4 28 64 10 17 C6 61 71 2F 90 25 26 E0 C5 38 2A C6 4A 99 9E F8 3B 1C</p> <p>B1 24 9A DA 84 BD 8F A8 5C 9C 07 81 15 19 12 FD F6 32 DB 69 C3 B3 7B 76 84 15 70 C9 65 73 TA 7A F6 D7 A1 21 4C F0 63 06 42 37 TD F3 A3 B5 D0 85 EF 06 02 FD F1 6D 44 3C 55 AB BA 8A 0F 45 A6 5C C8 53 19 F6 A1 75 06 90 A2 26 2D A3 90</p> <p>9B AB 18 53 2C E9 0F 05 83 3D 0E 49 76 33 D8 4D 8E 72 29 2F FB 51 3B 42 5F 76 C4 AE 54 45 E6 91 8D 2B 45 F9 A1 65 C0 3A D7 99 F5 AE F1 9D E9 45 D3 A3 CD DB 7A 74 03 D0 ED 7F 6D 00 1D 20 2A FD 23 2A 7D BB 08 13 72 9B 38 A0 89 F4</p> <p>7A 36 23 01 54 1D F6 FB F9 09 92 32 D7 CIC 34 38 24</p> <p>extended_master_secret empty</p> <p>ec_point_formats uncompressed [0x0]</p> <p>status_request OCSP -Implicit Responder</p> <p>psk_key_exchange_modes 01 01</p> <p>renegotiation_info 00</p> <p>0xfe00 00 00 01 00 01 8B 00 20 08 F0 C0 F2 E4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 45 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15</p> <p>F6 5E AC B2 14 57 EC 00 FB 93 90 2C 7E 88 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B BE 65 52 TA 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E</p> <p>2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 BE FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 C</p> <p>E4 C8 A4 D5 F3 3C D6 8F 9A DF C3 A0 9F 27 E4 C6 05 53 5F 83 38 B9 5A</p> <p>server_name www.higginbotham.com</p> <p>osrease [0x3a3a] 00</p> |
|--|--|

*Source: Fiddler Capture*

SignedCertTimestamp (RFC6962) empty  
ALPN h2, http/1.1  
signature\_algs ecdsa\_secp256r1\_sha256\_rsa\_pss\_rsae\_sha256\_rsa\_pkcs1\_sha256\_ecdsa\_secp384r1\_sha384\_rsa\_pss\_rsae\_sha384\_rsa\_pkcs1\_sha384\_rsa\_pss\_rsae\_sha512\_rsa\_pkcs1\_sha512  
0x001b 02 00 02  
supported\_groups grease [0x5a5a], unknown [0x6399], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]  
key\_share 04 ED 5A 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 32 5D 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85  
7A 95 44 33 3F C9 45 46 08 B4 C1 22 E1 9C 87 71 A3 5B C3 72 B0 E1 C9 D5 CA 9D C4 6B 80 05 5B 9D E2 A4 2B 36 F5 FB 82 67 9C C6 65 5F 82 3E B2 F0 BB F5 3B AF 08 11 A4 1C 00 5A 46 69 D9 57 AF 31 C6 C2 F4 F0 38 F1  
EC 97 6A 21 78 C4 A8 AA BA 85 CE C9 85 67 D7 T7,0C EB AB 79 1D 83 A3 B2 3B 60 D6 A3 C2 01 87 A7 F7 CC BB A0 E5 1E 1F 55 88 34 65 2F BA CA 06 C9 3B 17 9D A6 11 B2 39 9F 3C 00 6C BD D0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B 8E 2F  
C0 61 D0 54 55 73 99 23 8C 39 A3 B0 C6 73 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A B5 2C 21 80 CA 1F E2 AC B4 73 A5 FD E8 7B 6D B5 93 D6 10 37 A1 E3 C8 18 C6 49 D3 71 0F E0 75 BA 39 AA 08 66 56 6D DC 3C 1E 5D 8A 7E 1A B6 2E F1  
23 C1 C3 B1 5E 05 13 61 7C 42 DC 1C 3F P0 FA 47 A3 00 28 E6 A7 C1 26 7B D7 FB 9F E9 09 14 5A E3 B6 5A 5B 9B D2 D5 47 0D 6B 7B 8A 4A 71 6D 7E 01 37 F8 E0 A3 1E 58 6B D6 D2 C0 26 53 B8 98 80 09 DF 5A 20 22 E9 A3 DD D0 09 98 F0 4C A0  
11 99 AD C0 48 DA D1 37 3B 2A AF 72 43 91 BD 66 E6 23 56 C7 5A E0 5E 74 33 BF 3A 86 40 07 F3 7D 85 B1 38 70 23 05 30 1B 5E A5 C7 07 13 84 A9 90 B3 66 B5 11 A9 A9 61 72 3C 15 DC EC 94 3E 55 93 83 64 C0 BF 79 2F C3 22 6C B1 C4 AB 5B 12  
83 A0 B0 68 2B 98 2F 47 47 7B 2D 14 53 0B 43 A0 B2 88 C9 86 93 A7 E0 08 2D 4B 10 BA 70 A3 93 D1 75 07 59 C9 80 55 26 AB 84 90 C9 91 C7 92 10 B8 92 89 E3 A3 7F 03 73 FA 81 9F 99 BF 77 07 A1  
08 BB 58 48 CC 2D C1 FB BB 77 23 89 CB F0 92 BB 68 65 B3 E4 6C 68 59 20 88 B5 53 12 EB 49 EB A3 13 B3 78 A5 F1 A3 70 86 88 86 96 7F 23 9C 37 19 0C C0 3C 08 14 4B 67 78 4A F5 03 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 A4 AB  
CA 6B 3C 43 81 75 27 C9 7D 9B 88 4D 4B 39 6D AB 00 92 52 17 C3 C3 AD 10 A0 DE 0A B5 88 C7 49 5D 94 3A C9 35 95 02 2C 1E 72 3A 6D 26 16 51 54 31 68 1D 8A 72 9C 02 6F CC 86 91 E7 98  
2B 72 01 7B C4 34 07 61 05 82 53 94 6E 2C 30 74 TA A7 08 0D 91 77 TA 15 4C 5D F4 E6 88 64 9D 20 34 22 4F 29 2B 29 37 42 25 05 53 92 FD CA D9 13 8A 44 2A 23 98 EC 22 E9 01 24 6D 29 B8 DC 0B 80 45 B9 45 88 43 BA E4 99 3A 32 32 32 6C 66  
31 26 FC 06 97 C2 2E 7D 39 B9 AA 17 69 3C 94 66 FA C3 3B C9 E4 5C 75 BC 59 88 60 14 A7 26 83 FF 72 2C 0D FT 4F E9 88 80 33 67 6D F3 C2 C0 1A 5B 69 2F 61 48 53 94 9A FC 67 00 03 7A 8C 8E CB 4E 4C 2C 49 E0 82 8C DA B8 AF DA 6A 6D B5 08  
9D CA A9 A5 5A E2 91 50 5A 3C 0D B6 13 2C 0D CE 61 A8 4F E4 54 15 8B 3C A3 21 21 D8 28 CD AC 99 09 A2 37 B4 46 32 2F AF E4 AF E8 25 53 CC FC 4F 4B 78 A4 2F 28 64 10 17 C6 61 71 2F 90 25 6E C3 38 2A C4 6C A4 99 9E F8 3D 8C 1B  
48 CO AE DC B0 86 1C 85 B3 75 4F 4B 5A 14 3F C0 90 CD E6 B7 18 12 6E 34 52 B1 C3 A1 B4 17 72 F8 25 20 88 21 6B B1 AC 3C A4 4E 23 58 CC 0A 82 32 07 76 C5 23 8C C0 40 07 00 C0 B0 C2 4A 48 D3 B7 13 00 11 C5 88 73 26 F2 EA 96 15 55  
CC 8C 8A 8D 69 93 B2 8C 1E 43 34 23 C8 57 4B 65 2F E6 29 97 C9 39 39 BD E7 07 0F 2D C2 AF 53 DA AA 56 C1 D7 4B 93 54 D9 67 C5 D9 F5 1E 7B 86 49 57 45 85 58 56 84 32 42 B5 39 60 62 40 C2 96  
B1 24 9A DA B4 8D FD 8A 9C 5C C0 87 1E 19 12 FD FA 32 DB 69 C3 B3 7B 76 84 15 70 C9 69 73 TA F6 A6 D7 A1 21 4C F0 63 06 42 37 TD F3 3A B5 D0 85 EF D6 02 FD F1 6D 4A 3C 55 AB BA 0F 45 A6 5C C8 53 19 F6 A1 75 06 90 A0 26 2D A3 90  
9B AB 18 53 2C E9 0F 05 83 3D DE 49 76 33 D8 4D 8E 72 29 2F FB 51 3B 42 5F 76 C4 AE 54 45 E6 91 8D 2B 45 F0 9A 1E 65 C0 3A D7 99 F5 AE F1 9D E9 45 D3 A3 CD DB 7A 74 E0 3D ED F7 6D 00 1D 00 20 FA D2 73 2A 7D BB 08 13 72 9B 38 A0 89 F4  
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 C6 34 38 24  
extended\_master\_secret empty  
ec\_point\_formats uncompressed [0x0]  
status\_request OCSP - Implicit Responder  
psk\_key\_exchange\_modes 01 01  
renegotiation\_info 00  
0xfe0d 00 00 01 00 01 0B 00 20 08 F0 C0 F2 E4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15  
F6 5E AC B2 14 57 6C 00 FB F9 30 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B BE 66 B2 52 TA 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E  
2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 BE FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 C  
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A  
server\_name www.higginbotham.com  
grease [0x3a3a] 00

The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

All handshake messages after the ServerHello are now encrypted. The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

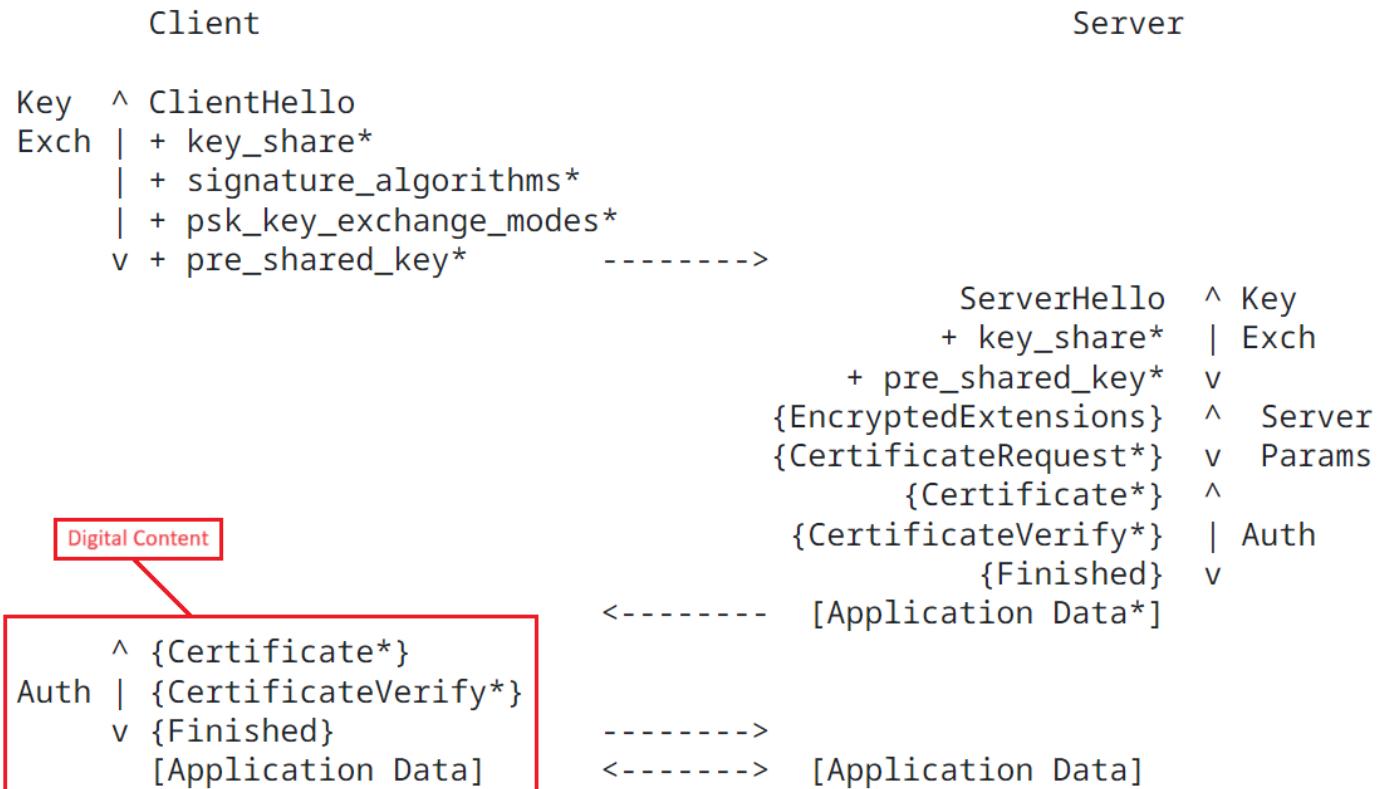
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.  
First encryption
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message ([Section 4.3.2](#)). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

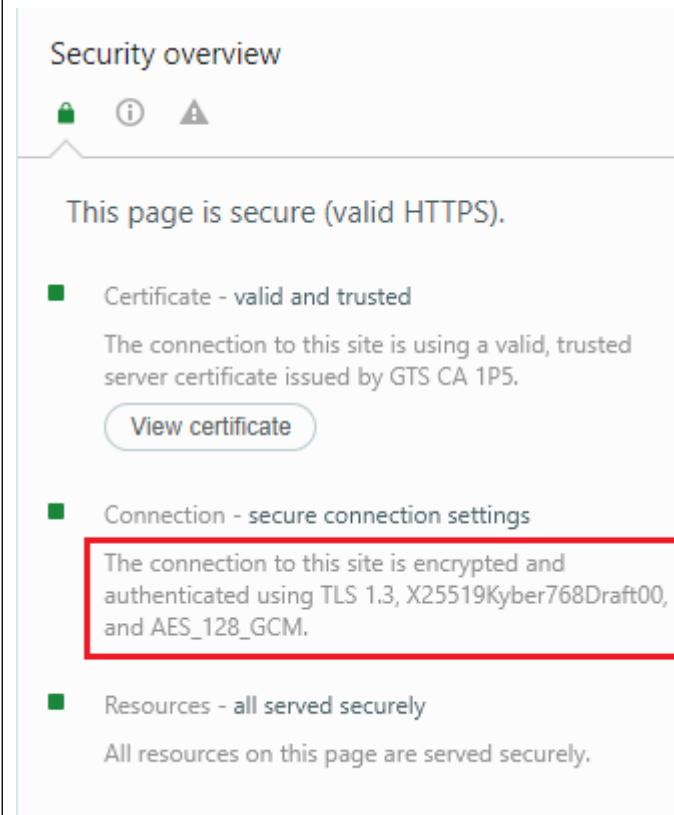
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |  |
|---|--|
|   | <p><b>Introduction</b></p> <p>The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream. Specifically, the secure channel should provide the following properties:</p> <ul style="list-style-type: none"> <li>- Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated.</li> </ul> <p style="border: 1px solid red; padding: 5px;">Authentication can happen via asymmetric cryptography (e.g., RSA [RSA], the Elliptic Curve Digital Signature Algorithm (ECDSA) [ECDSA], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032]) or a symmetric pre-shared key (PSK).</p> <ul style="list-style-type: none"> <li>- Confidentiality: Data sent over the channel after establishment is only visible to the endpoints. TLS does not hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.</li> <li>- Integrity: Data sent over the channel after establishment cannot be modified by attackers without detection.</li> </ul> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446">https://datatracker.ietf.org/doc/html/rfc8446</a></p> |
| associating a first decryption algorithm with the encrypted bit stream; | <p>The standard practices associating a first decryption algorithm (e.g., signature decryption algorithm i.e., SHA256RSA decryption algorithm) with the encrypted bit stream (e.g., encrypted certificate with signature encryption algorithm).</p> <p>The standard practices providing a two-level encryption security for data</p>   |

communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA encryption algorithm) and generates a ciphertext.

The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it. The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate.

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

As shown below, the accused instrumentality discloses the signature decryption algorithm.

[Thumbprint]  
129157F7A5852AD23DF17FE2934E6396DB6B0251

[Signature Algorithm]  
sha256RSA(1.2.840.113549.1.1.11)

[Public Key]  
Algorithm: RSA  
Length: 2048  
Key Blob: 30 82 01 0a 02 82 01 01 00 ca b8 66 5f 8a 37 94 14 e1 2f e3 49 cf d7 ad cb 20 8c 54 93 6b af 66 56 02 11 26 db 20 13 00 a0 ce fe ae e9 ac 8d 68 8f a0 56 fd ff b4 b1 4b 9c 54 ff 55 4a 95 76 b9 db 3c 77 5e 9a b1 dd 30 74 36 77 cb 92 12 df 17 62 6d 2a aa 74 3c f6 68 c8 34 8d eb 69 97 9e f2 43 02 fa d7 76 99 c0 bc b3 56 2d 17 ca 39 c8 00 5d 91 c8 1d 9b c9 70 c8 c1 b0 da 40 23 3e 9a fe a9 ed e4 60 ce 15 b0 af 43 a3 07 a5 81 25 dd c3 92 e7 72 9e 27 df bd 01 05 2a b4 60 b8 83 6b fa bc db 97 51 47 a8 2e d0 52 7f c2 0a 1c 02 96 d2 c7 fd 04 a5 7e 78 04 ee 59 ed 33 9a e4 cc 42 bb 87 36 95 ac 4b 9b 6b 4f 96 4c ed 20 f7 bd 71 4b bc d8 91 81 3c be 56 c8 05 a6 e1 7f 66 85 88 46 91 96 a3 e4 e8 bb 45 f5 b8 78 25 4b 02 cb 23 e6 54 94 ed 94 f6 87 3d 29 55 31 eb 59 e9 a1 54 4c 50 b0 74 2c a7 b6 1e 7f 23 02 03 01 00 01  
Parameters: 05 00

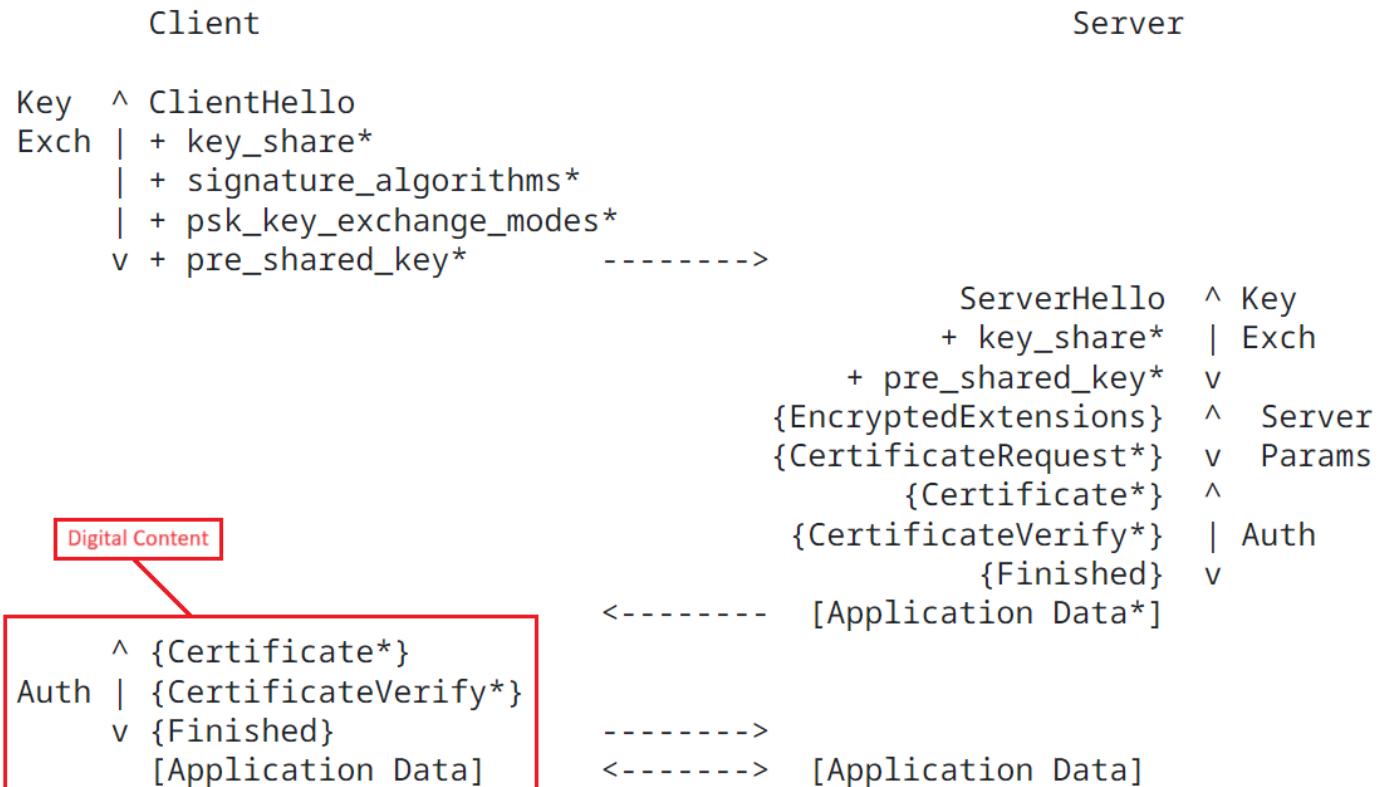
[Extensions]  
\* Key Usage(2.5.29.15):  
Digital Signature, Key Encipherment (a0)

*Source: Fiddler Capture*

First decryption  
algorithm

| OID description   |   |
|---|---|
|   | <b>First decryption algorithm identifier</b>  |
| <b>OID:</b>   | {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1)}<br>sha256WithRSAEncryption(11)<br>1.2.840.113549.1.1.11<br>/ISO/Member-Body/US/113549/1/1/11       |
|   | (ASN.1 notation)<br>(dot notation)<br>(OID-IRI notation)  |
| <b>Description:</b>   | Public-Key Cryptography Standards (PKCS) #1 version 1.5 signature algorithm with Secure Hash Algorithm 256 (SHA256) and Rivest, Shamir and Adleman (RSA) encryption |
| <a href="http://oid-info.com/get/1.2.840.113549.1.1.11">http://oid-info.com/get/1.2.840.113549.1.1.11</a>                     |   |
| <pre>-- When the following OIDs are used in an AlgorithmIdentifier, the -- parameters MUST be present and MUST be NULL.</pre> |   |
| <pre>sha224WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 14 }</pre>  |   |
| <pre><u>sha256WithRSAEncryption</u> OBJECT IDENTIFIER ::= { pkcs-1 11 }</pre>   |   |
| <pre>sha384WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 12 }</pre>  |   |
| <pre>sha512WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 13 }</pre>  |   |
| <a href="https://www.ietf.org/rfc/rfc4055.txt">https://www.ietf.org/rfc/rfc4055.txt</a>                                       |   |

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

All handshake messages after the ServerHello are now encrypted. The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

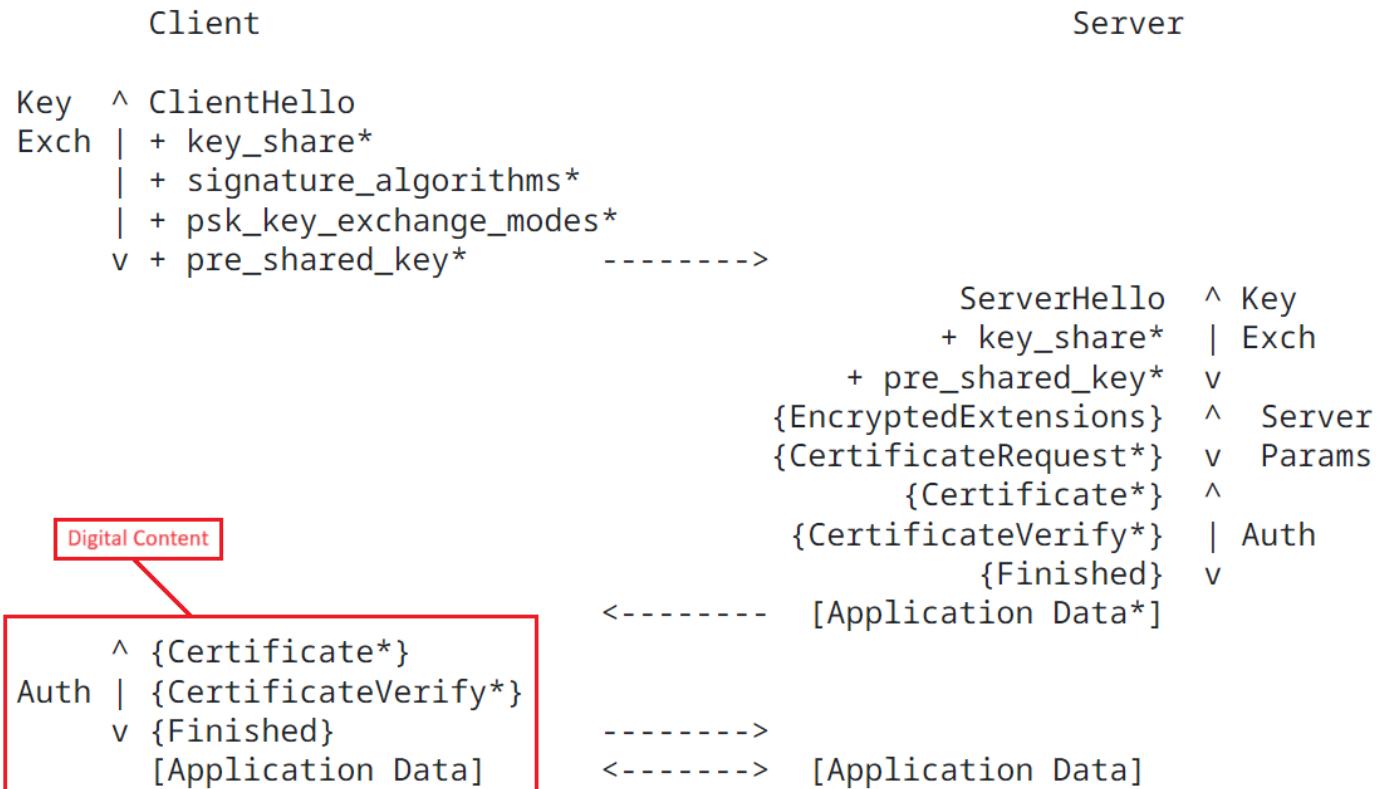
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.  
First encryption
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message ([Section 4.3.2](#)). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.3. Certificate Verify

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

First  
decryption  
algorithm  
information

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

certificate\_request\_context: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The certificate\_request\_context MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

extensions: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

As shown below, the receiving party will be able to decrypt the encrypted message with the provided signature decryption algorithm information i.e., SHA-256 RSA decryption algorithm.

We are now prepared to show that we can decrypt encrypted messages. We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

|  |  |
|--|--|
|  | <p>The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A <i>cryptographic hash function</i> is a function that computes a <i>message authentication code</i> from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that <math>H</math> is a cryptographic hash function. To sign a message <math>m</math>, party A computes <math>h = D_A(H(m))</math> and sends <math>E_B(m, h)</math> to B. Party B now has evidence that A signed <math>m</math> because <math>E_A(h) = H(m)</math>, and A is the only one who could have generated a value <math>h</math> with that property.</p> |
| encrypting both the encrypted bit stream and the first decryption algorithm with a second encryption algorithm to yield a second bit stream; | <p>The standard practices encrypting both the encrypted bit stream (e.g., encrypted digital certificate) and the first decryption algorithm (e.g., signature decryption algorithm) with a second encryption algorithm (e.g., cipher suit selected from one of the AEAD algorithms such as TLS_AES_256_GCM_SHA384, etc.) to yield a second bit stream (e.g., TLS ciphertext bitstream).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA algorithm) and generates a ciphertext.</p> <p>The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it.</p>                           |

The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate. The standard further practices encrypting the authentication message, including the encrypted digital certification and the certificate verify message, with a second decryption algorithm i.e., AEAD algorithm such as TLS\_AES\_256\_GCM\_SHA384, etc.

#### Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.

[View certificate](#)

- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.

- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

| Headers   | TextView   | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm |
|-----------|--|------------|----------|---------|------|---------|-----|------|-----|-----------------------------|
| 00000019  | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 |            |          |         |      |         |     |      |     |                             |
| 00000032  | 77 77 2E 68 69 67 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A |            |          |         |      |         |     |      |     |                             |
| 0000004B  | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55 |            |          |         |      |         |     |      |     |                             |
| 00000064  | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 |            |          |         |      |         |     |      |     |                             |
| 0000007D  | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36 |            |          |         |      |         |     |      |     |                             |
| 00000096  | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48 |            |          |         |      |         |     |      |     |                             |
| 000000AF  | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31 |            |          |         |      |         |     |      |     |                             |
| 000000C8  | 32 36 2E 30 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D |            |          |         |      |         |     |      |     |                             |
| 000000E1  | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E |            |          |         |      |         |     |      |     |                             |
| 000000FA  | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E |            |          |         |      |         |     |      |     |                             |
| 00000013  | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20 |            |          |         |      |         |     |      |     |                             |
| 00000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65 |            |          |         |      |         |     |      |     |                             |
| 00000145  | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72 |            |          |         |      |         |     |      |     |                             |
| 0000015E  | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53 |            |          |         |      |         |     |      |     |                             |
| 00000177  | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69 |            |          |         |      |         |     |      |     |                             |
| 00000190  | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A |            |          |         |      |         |     |      |     |                             |
| 000001A9  | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20 |            |          |         |      |         |     |      |     |                             |
| 000001C2  | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30 |            |          |         |      |         |     |      |     |                             |
| 000001DB  | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37 |            |          |         |      |         |     |      |     |                             |
| 000001F4  | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69 |            |          |         |      |         |     |      |     |                             |
| 0000020D  | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A |            |          |         |      |         |     |      |     |                             |
| 00000226  | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32 |            |          |         |      |         |     |      |     |                             |
| 0000023F  | 20 34 44 20 32 43 20 43 42 20 36 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |            |          |         |      |         |     |      |     |                             |

Source: Fiddler Capture

|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 39 2C 7B 89 AA BB C5 AF E |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | 0C 2A 73 3E B4 85 7A 95   |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |

*Source: Fiddler Capture*

The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm. The authentication message is a TLS plaintext.

handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [\[RFC5116\]](#), [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

All handshake messages after the ServerHello are now encrypted.  
The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

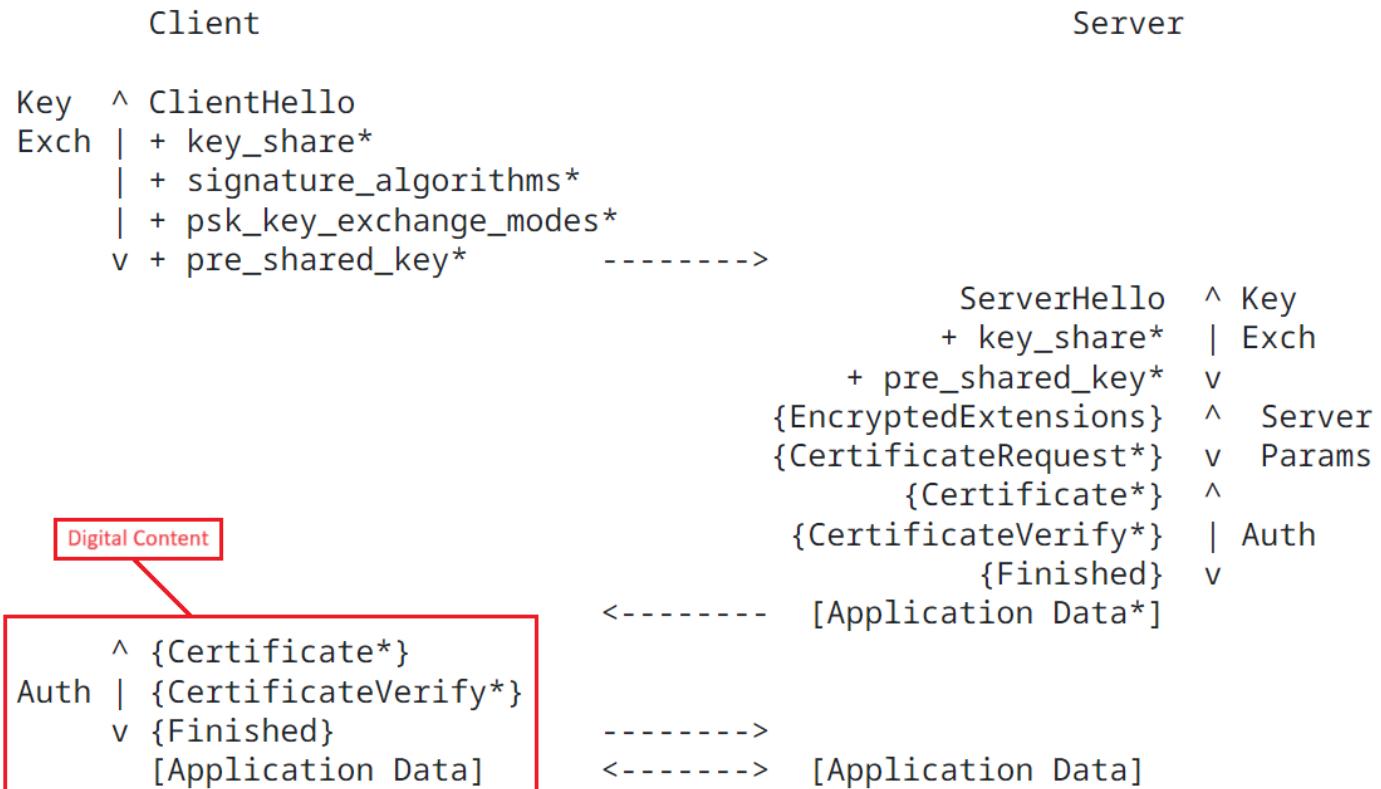
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.3. Certificate Verify

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

First  
decryption  
algorithm  
information

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |   |
|---|---|
|   | <p>The <u>"extension_data"</u> field of these extensions contains a <u>SignatureSchemeList</u> value:</p> <pre> enum {     /* RSASSA-PKCS1-v1_5 algorithms */     rsa_pkcs1_sha256(0x0401),     rsa_pkcs1_sha384(0x0501),     rsa_pkcs1_sha512(0x0601),      /* ECDSA algorithms */     ecdsa_secp256r1_sha256(0x0403),     ecdsa_secp384r1_sha384(0x0503),     ecdsa_secp521r1_sha512(0x0603),      /* RSASSA-PSS algorithms with public key OID rsaEncryption */     rsa_pss_rsae_sha256(0x0804),     rsa_pss_rsae_sha384(0x0805),     rsa_pss_rsae_sha512(0x0806),      /* EdDSA algorithms */     ed25519(0x0807),     ed448(0x0808),      /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */     rsa_pss_pss_sha256(0x0809),     rsa_pss_pss_sha384(0x080a),     rsa_pss_pss_sha512(0x080b), </pre> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| associating a second decryption algorithm with the second bit stream. | <p>The standard practices associating a second decryption algorithm (e.g., cipher suit selected from one of the AEAD algorithms such as TLS_AES_256_GCM_SHA384, etc.) with the second bit stream (e.g., TLS ciphertext bitstream).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA algorithm) and generates a ciphertext.</p>   |

The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it. The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate. The standard further practices encrypting the authentication message, including the encrypted digital certification and the certificate verify message, with a second decryption algorithm i.e., AEAD algorithm such as TLS\_AES\_256\_GCM\_SHA384, etc.

#### Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.

[View certificate](#)

- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.

- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

The screenshot shows a Fiddler network traffic capture. On the left, a list of requests is displayed, mostly in green (HTTPS) and some in blue (HTTP). Request 8 is highlighted with a red box and labeled "Tunnel To www.higginbotham.com:443". The right pane shows a detailed view of this session. It includes tabs for Transformer, Headers, TextView, SyntaxView, ImageView, HexView, WebView, Auth, Caching, Cookies, Raw, JSON, and XML. The Headers tab shows the secure connection details: "Secure Protocol: TLS 1.3" and "Cipher Suite: TLS\_AES\_256\_GCM\_SHA384". The TextView tab displays the certificate information, which is also highlighted with a red box. The certificate details are as follows:

|  |
|--|
| == Server Certificate =====                      |
| [Version]  |
| V3   |
| [Subject]  |
| CN=higginbotham.com                              |
| Simple Name: higginbotham.com                    |
| DNS Name: higginbotham.com                       |
| [Issuer]   |
| CN=GTS CA 1P5, O=Google Trust Services LLC, C=US |
| Simple Name: GTS CA 1P5                          |
| DNS Name: GTS CA 1P5                             |

*Source: Fiddler Capture*

|          | <p>Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.</p> <p>Secure Protocol: TLS 1.3<br/>Cipher Suite: TLS_AES_256_GCM_SHA384</p> <pre>-- Server Certificate ----- [Version] V3  [Subject] CN=higginbotham.com Simple Name: higginbotham.com DNS Name: higginbotham.com  [Issuer] CN=GTS CA 1P5, O=Google Trust Services LLC, C=US</pre> <p><i>Source: Fiddler Capture</i></p>   |            |          |          |            |          |         |      |         |                           |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
|----------|--|------------|----------|----------|------------|----------|---------|------|---------|---------------------------|------|-----|------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|--------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|
|          | <table border="1"> <thead> <tr> <th>Headers</th><th>TextView</th><th>SyntaxView</th><th>WebForms</th><th>HexView</th><th>Auth</th><th>Cookies</th><th>Raw</th><th>JSON</th><th>XML</th><th>Second bitstream</th></tr> </thead> <tbody> <tr><td>000004C9</td><td>36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>67 98 9F 8B 88 D9 A0 C6 1</td></tr> <tr><td>000004E2</td><td>35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>5 CC 37 BB 51 91 1D 30 CE</td></tr> <tr><td>000004FB</td><td>20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>8D F6 8E 5F 54 C2 5A E3</td></tr> <tr><td>00000514</td><td>38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>9 2C 7B 89 AA BB C5 AF E</td></tr> <tr><td>0000052D</td><td>39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>9 56 10 B8 15 57 94 A1 A1</td></tr> <tr><td>00000546</td><td>20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>0C 2A 73 3E B4 85 7A 95</td></tr> <tr><td>0000055F</td><td>34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>44 34 07 19 C8 33 3F EC 9</td></tr> <tr><td>00000578</td><td>43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>C 45 46 0B 84 C9 1C 22 E1</td></tr> <tr><td>00000591</td><td>20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>3C 9F 87 71 A3 5B C3 72</td></tr> <tr><td>000005AA</td><td>42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>B0 E1 C9 D5 CA 9D C4 69 B</td></tr> <tr><td>000005C3</td><td>30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>0 05 5B 9D E2 A4 2B 56 F5</td></tr> <tr><td>000005DC</td><td>20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>BF 82 84 67 9C C6 65 5F</td></tr> <tr><td>000005F5</td><td>38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>82 3E B2 F0 BB F5 3B AF 0</td></tr> <tr><td>0000060E</td><td>46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>F 86 11 A4 1C 00 5A 46 69</td></tr> <tr><td>00000627</td><td>20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D9 57 AF 31 C6 C2 F4 F0</td></tr> <tr><td>00000640</td><td>33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>38 F7 EC 97 6A 21 78 C4 A</td></tr> <tr><td>00000659</td><td>38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>8 AA B4 85 CE C9 85 67 D7</td></tr> <tr><td>00000672</td><td>20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>7A 0C EB AB 79 1D 83 3A</td></tr> <tr><td>000006BB</td><td>42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>B2 3B 60 D6 A3 C2 01 87 A</td></tr> <tr><td>000006A4</td><td>37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>7 F7 CC B8 A0 E5 1E 1F 55</td></tr> <tr><td>000006BD</td><td>20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>88 34 65 2F BA CA 06 9C</td></tr> <tr><td>000006D6</td><td>33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>3B 17 9D A6 11 B2 39 9F 3</td></tr> <tr><td>000006EF</td><td>34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>4 00 6C CB D0 53 3E 57 16</td></tr> </tbody> </table> <p><i>Source: Fiddler Capture</i></p> <p>The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS</p> |            | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw                       | JSON | XML | Second bitstream | 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |  |  |  |  |  |  |  |  | 67 98 9F 8B 88 D9 A0 C6 1 | 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |  |  |  |  |  |  |  |  | 5 CC 37 BB 51 91 1D 30 CE | 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |  |  |  |  |  |  |  |  | 8D F6 8E 5F 54 C2 5A E3 | 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |  |  |  |  |  |  |  |  | 9 2C 7B 89 AA BB C5 AF E | 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |  |  |  |  |  |  |  |  | 9 56 10 B8 15 57 94 A1 A1 | 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |  |  |  |  |  |  |  |  | 0C 2A 73 3E B4 85 7A 95 | 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |  |  |  |  |  |  |  |  | 44 34 07 19 C8 33 3F EC 9 | 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |  |  |  |  |  |  |  |  | C 45 46 0B 84 C9 1C 22 E1 | 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |  |  |  |  |  |  |  |  | 3C 9F 87 71 A3 5B C3 72 | 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |  |  |  |  |  |  |  |  | B0 E1 C9 D5 CA 9D C4 69 B | 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |  |  |  |  |  |  |  |  | 0 05 5B 9D E2 A4 2B 56 F5 | 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |  |  |  |  |  |  |  |  | BF 82 84 67 9C C6 65 5F | 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |  |  |  |  |  |  |  |  | 82 3E B2 F0 BB F5 3B AF 0 | 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |  |  |  |  |  |  |  |  | F 86 11 A4 1C 00 5A 46 69 | 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |  |  |  |  |  |  |  |  | D9 57 AF 31 C6 C2 F4 F0 | 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |  |  |  |  |  |  |  |  | 38 F7 EC 97 6A 21 78 C4 A | 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |  |  |  |  |  |  |  |  | 8 AA B4 85 CE C9 85 67 D7 | 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |  |  |  |  |  |  |  |  | 7A 0C EB AB 79 1D 83 3A | 000006BB | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |  |  |  |  |  |  |  |  | B2 3B 60 D6 A3 C2 01 87 A | 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |  |  |  |  |  |  |  |  | 7 F7 CC B8 A0 E5 1E 1F 55 | 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |  |  |  |  |  |  |  |  | 88 34 65 2F BA CA 06 9C | 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |  |  |  |  |  |  |  |  | 3B 17 9D A6 11 B2 39 9F 3 | 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |  |  |  |  |  |  |  |  | 4 00 6C CB D0 53 3E 57 16 |
| Headers  | TextView   | SyntaxView | WebForms | HexView  | Auth       | Cookies  | Raw     | JSON | XML     | Second bitstream          |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31   |            |          |          |            |          |         |      |         | 67 98 9F 8B 88 D9 A0 C6 1 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45   |            |          |          |            |          |         |      |         | 5 CC 37 BB 51 91 1D 30 CE |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20   |            |          |          |            |          |         |      |         | 8D F6 8E 5F 54 C2 5A E3   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45   |            |          |          |            |          |         |      |         | 9 2C 7B 89 AA BB C5 AF E  |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31   |            |          |          |            |          |         |      |         | 9 56 10 B8 15 57 94 A1 A1 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20   |            |          |          |            |          |         |      |         | 0C 2A 73 3E B4 85 7A 95   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39   |            |          |          |            |          |         |      |         | 44 34 07 19 C8 33 3F EC 9 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31   |            |          |          |            |          |         |      |         | C 45 46 0B 84 C9 1C 22 E1 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20   |            |          |          |            |          |         |      |         | 3C 9F 87 71 A3 5B C3 72   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42   |            |          |          |            |          |         |      |         | B0 E1 C9 D5 CA 9D C4 69 B |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35   |            |          |          |            |          |         |      |         | 0 05 5B 9D E2 A4 2B 56 F5 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20   |            |          |          |            |          |         |      |         | BF 82 84 67 9C C6 65 5F   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30   |            |          |          |            |          |         |      |         | 82 3E B2 F0 BB F5 3B AF 0 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39   |            |          |          |            |          |         |      |         | F 86 11 A4 1C 00 5A 46 69 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20   |            |          |          |            |          |         |      |         | D9 57 AF 31 C6 C2 F4 F0   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41   |            |          |          |            |          |         |      |         | 38 F7 EC 97 6A 21 78 C4 A |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37   |            |          |          |            |          |         |      |         | 8 AA B4 85 CE C9 85 67 D7 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20   |            |          |          |            |          |         |      |         | 7A 0C EB AB 79 1D 83 3A   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006BB | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41   |            |          |          |            |          |         |      |         | B2 3B 60 D6 A3 C2 01 87 A |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35   |            |          |          |            |          |         |      |         | 7 F7 CC B8 A0 E5 1E 1F 55 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20   |            |          |          |            |          |         |      |         | 88 34 65 2F BA CA 06 9C   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33   |            |          |          |            |          |         |      |         | 3B 17 9D A6 11 B2 39 9F 3 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36   |            |          |          |            |          |         |      |         | 4 00 6C CB D0 53 3E 57 16 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |

communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

Further, the AEAD encrypted message comprises a ciphertext (e.g., encrypted ciphertext after the encryption by the second encryption algorithm), nonce (e.g., associating second decryption algo), key and associated data. The maximum length of nonce is a cipher suit specific element. The nonce and associated data are utilized in decryption of the AEAD encrypted message.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.2. Authenticated Decryption

Second decryption algorithm

The authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic. A ciphertext C, a nonce N, and associated data A are authentic for key K when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [RFC5116].. [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:  
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

All handshake messages after the ServerHello are now encrypted.  
The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

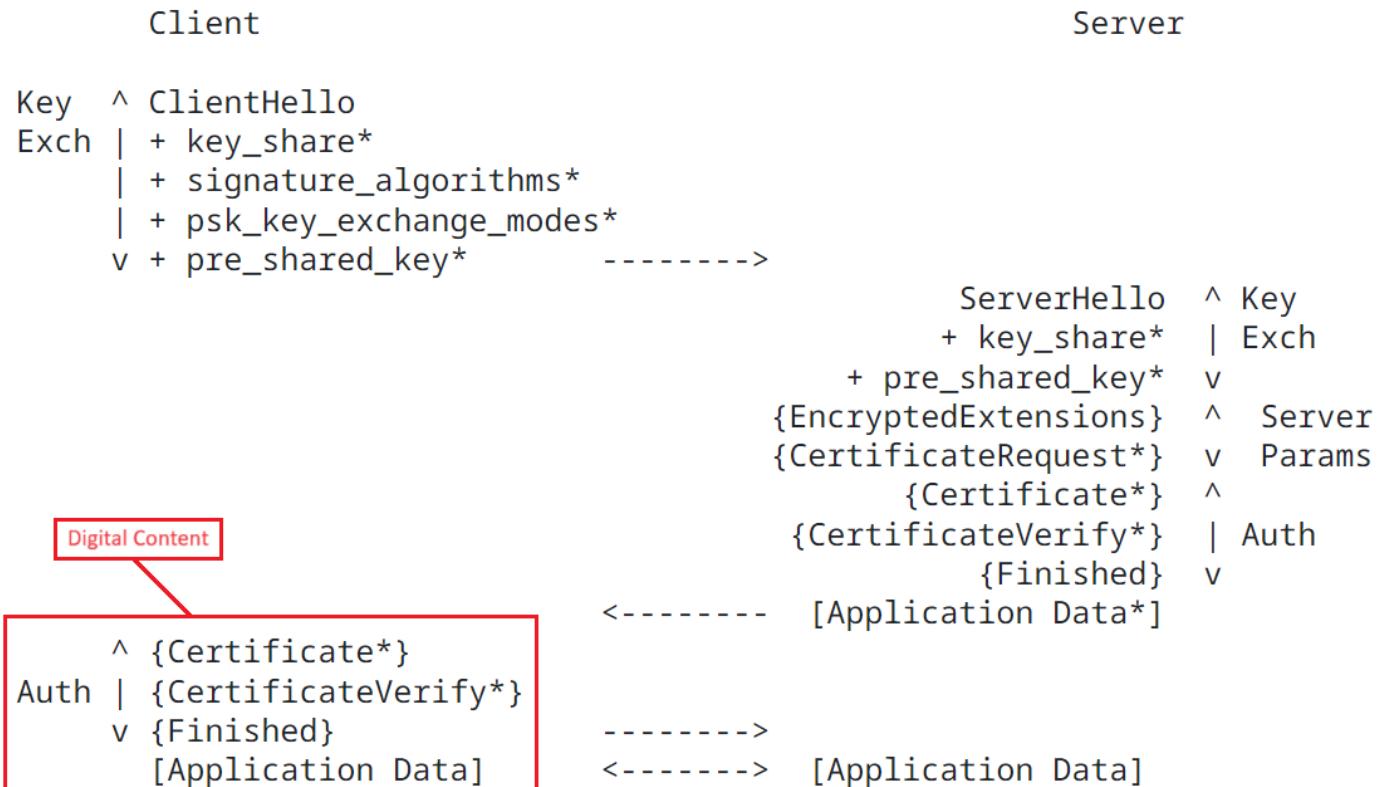
#### **4.4. Authentication Messages**

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block.

These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |  |
|---|--|
|   | <p>The <u>"extension_data"</u> field of these extensions contains a <u>SignatureSchemeList</u> value:</p> <pre>enum {     /* RSASSA-PKCS1-v1_5 algorithms */     rsa_pkcs1_sha256(0x0401),     rsa_pkcs1_sha384(0x0501),     rsa_pkcs1_sha512(0x0601),      /* ECDSA algorithms */     ecdsa_secp256r1_sha256(0x0403),     ecdsa_secp384r1_sha384(0x0503),     ecdsa_secp521r1_sha512(0x0603),      /* RSASSA-PSS algorithms with public key OID rsaEncryption */     rsa_pss_rsae_sha256(0x0804),     rsa_pss_rsae_sha384(0x0805),     rsa_pss_rsae_sha512(0x0806),      /* EdDSA algorithms */     ed25519(0x0807),     ed448(0x0808),      /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */     rsa_pss_pss_sha256(0x0809),     rsa_pss_pss_sha384(0x080a),     rsa_pss_pss_sha512(0x080b), <a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></pre> |
| 2. The method of claim 1, further comprising decrypting the first bit stream and the second | The standard further discloses decrypting the first bit stream (e.g., encrypted digital certificate with signature encryption algorithm i.e., SHA-256 RSA, etc.) and the second bit stream (e.g., a second-level encryption with AEAD encryption algorithm such as TLS AES 256 GCM SHA384, etc.) with the first associated decryption  |

|   |   |
|---|---|
| bit stream with the first associated decryption algorithm and the second associated decryption algorithm wherein the decryption is accomplished by a target unit. | <p>algorithm (e.g., signature decryption algorithm i.e., SHA-256 RSA, etc.) and the second associated decryption algorithm (e.g., cipher suit selected from one of the AEAD decryption algorithms such as TLS_AES_256_GCM_SHA384, etc.) wherein the decryption is accomplished by a target unit (e.g., a server of the accused instrumentality).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA algorithm) and generates a ciphertext.</p> <p>The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it. The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate. The standard further practices encrypting the authentication message, including the encrypted digital certification and the certificate verify message, with a second decryption algorithm i.e., AEAD algorithm such as TLS_AES_256_GCM_SHA384, etc.</p> |
|---|---|

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

The screenshot shows a Fiddler network traffic capture. On the left, a list of requests is displayed, showing various HTTP and HTTPS connections between the user's browser and various websites like www.higginbotham.com, www.googleapis.com, and cdnjs.cloudflare.com. Request 8 is highlighted with a red box and labeled "Tunnel to www.higginbotham.com:443". On the right, a detailed SSL/TLS analysis is shown for this connection. It includes tabs for Transformer, Headers, TextView, SyntaxView, ImageView, HexView, WebView, Auth, Caching, Cookies, Raw, JSON, and XML. The Headers tab shows the secure protocol as "TLS 1.3" and the cipher suite as "TLS\_AES\_256\_GCM\_SHA384". The Transformer tab shows the server certificate details, including the subject (CN=higginbotham.com), version (V3), and issuer (CN=GTS CA 1P5, O=Google Trust Services LLC, C=US). The Headers tab also lists the simple name and DNS name as higginbotham.com.

*Source: Fiddler Capture*

|  |  |
|--|--|
|  | <pre> SignedCertTimestamp (RFC5962) empty ALPN h2, http/1.1 signature_algs ecdsa_secp256r1_sha256,rsa_pss_rsae_sha256,rsa_pkcs1_sha256,ecdsa_secp384r1_sha384,rsa_pss_rsae_sha384,rsa_pkcs1_sha384,rsa_pss_rsae_sha512,rsa_pkcs1_sha512 0x001b 02 00 02 supported_groups grease [0x5a5a], unknown [0x6399], x25519 [0xd1], secp256r1 [0x17], secp384r1 [0x18] key_share 04 ED 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 32 5D 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85 7A 95 44 34 07 19 C8 33 3F EC 9C 45 46 08 84 C9 1C 22 E1 3C 9F 87 71 A3 5B C3 72 80 E1 C9 D5 CA 9D C4 69 B0 05 5B 9D E2 A4 2B 56 F5 BF 82 84 67 9C C6 65 5F 82 3E B2 F0 B8 F5 3B AF 0F 86 11 A4 1C 00 5A 46 68 D9 57 AF 31 C6 C2 F4 F0 38 F1 EC 97 6A 21 78 C4 A8 AA B4 85 CE C9 85 67 D7 7A 0C EB AB 79 1D 83 3A B2 3B 60 D6 A3 C2 01 87 A7 F7 CC B8 A0 E5 1E 1F 55 88 34 65 2B BA CA 06 9C 3B 17 9D A6 11 B2 39 9F 34 00 6C CB D0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E 2F C0 61 1D A0 54 55 73 A1 99 23 8C 39 BA A0 C6 73 A8 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A 85 2C 21 80 CA 1F E2 AC B4 73 A5 FD EB 7B 6D 85 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F 0E 75 BA 39 AA 0B 66 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F1 11 99 AD 04 08 74 52 17 9D 7A 2F 43 23 56 C7 5A E0 5E 74 33 9F 2B 3A 0E 40 07 F7 D0 95 81 38 70 23 05 30 1B 5E A5 C7 07 13 84 84 A9 90 B3 66 5B 11 A9 A6 71 2C 15 DC EC 94 3E 55 93 83 64 C0 BF 79 2F C3 22 6C B1 C4 AB 5B 12 23 C1 C3 B1 E8 05 13 61 C7 42 DC 1C 3F F0 AB 47 A3 00 28 E6 A7 C1 C1 26 7B D7 FB 9F E9 09 14 5A E3 B4 5A 5B 9B D2 D5 47 0D 6B 7B 4A 71 6D 7E 01 37 F8 E0 A3 1E 58 6B D6 D2 C0 26 53 8B 9B 00 09 DF 5A 20 22 E9 A3 DD 00 9E 98 F0 4C A0 83 A0 B0 8F 26 98 2F 47 47 7B 2B 10 49 20 8F 4E DA 90 E3 A0 82 98 C8 89 6E 63 7A 8A AA A7 E6 C8 08 2D 4B 10 8A 70 A0 3A 93 D1 75 07 59 1C 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 99 3C E8 A3 A7 F0 73 D4 FA 81 9F 99 BF 70 B7 A1 08 BB 58 A8 C2 2D C1 FB BB 27 89 CB F0 92 BB A8 65 B3 E4 6C 6B 59 20 8B 55 13 2B 49 EB A3 13 B3 78 A5 F1 3A 70 86 8B 8B 96 7F 23 9C 37 19 0C C0 3C 0B 14 8F 67 78 4A F5 03 04 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 A4 AB CA 6B 3C 45 46 08 84 C9 1C 22 E1 3C 9F 87 71 A3 5B C3 72 80 E1 C9 D5 CA 9D C4 69 B0 05 5B 9D E2 A4 2B 56 F5 BF 82 84 67 9C C6 65 5F 82 3E B2 F0 B8 F5 3B AF 0F 86 11 A4 1C 00 5A 46 68 D9 57 AF 31 C6 C2 F4 F0 38 F1 83 A0 B0 8F 26 98 2F 47 47 7B 2B 10 49 20 8F 4E DA 90 E3 A0 82 98 C8 89 6E 63 7A 8A AA A7 E6 C8 08 2D 4B 10 8A 70 A0 3A 93 D1 75 07 59 1C 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 99 3C E8 A3 A7 F0 73 D4 FA 81 9F 99 BF 70 B7 A1 2B 72 01 7B C4 34 07 61 05 82 93 E4 6C 2C 30 74 A3 07 61 91 77 A4 14 5C 4D 4F E2 68 66 94 AD 20 34 22 4F 29 2B 37 42 25 08 93 F4 CD A9 19 3A 44 8A 23 98 EC 22 E9 01 24 6D 29 8B DC 08 B0 45 B9 45 88 43 BA E4 99 3A 32 23 32 66 31 26 FC 06 97 1C 2E 7D 39 BA A9 17 69 3C 94 66 FA C3 3B C9 E4 5C 75 BC 59 8B 60 14 A7 26 83 FF 72 2C 0D F7 44 2B 89 9F 8B 98 80 33 67 6D F3 C2 00 7A 8C 78 E5 92 8A F0 67 00 03 7A 8C 4E 4C 2C 49 4F 82 2C DA 88 AF DA 6A 6D B5 08 9D CA A9 45 5A 2</pre> |
|--|--|

[Thumbprint]  
129157F7A5852AD23DF17FE2934E6396DB6B0251

[Signature Algorithm]  
sha256RSA(1.2.840.113549.1.1.11)

First decryption  
algorithm

[Public Key]  
Algorithm: RSA  
Length: 2048

Key Blob: 30 82 01 0a 02 82 01 01 00 ca b8 66 5f 8a 37 94 14 e1 2f e3 49 cf d7 ad cb 20 8c 54 93 6b af 66 56 02 11 26 db 20 13 00 a0 ce fe ae e9 ac 8d 68 8f a0 56 fd ff b4 14b 9c 54 f5 95 76 b9 db 3c 77 5e 9a b1 dd 30 74 36 77 cb 92 12 df 17 62 6d 2a aa 74 3c f6 68 c8 34 8d eb 69 97 9e f2 43 02 fa d7 76 99 c0 bc b3 56 2d 17 ca 39 c8 00 5d 91 c8 1d 9b c9 70 c8 c1 b0 da 40 23 3e 9a fe a9 ed e4 60 ce 15 b0 af 43 a3 07 a5 81 25 dd c3 92 e7 72 9e 27 df bd 01 05 2a b4 60 b8 83 6b fa bc db 97 51 47 a8 2e d0 52 f7 c2 0a 1c 02 96 d2 c7 fd 04 a5 7e 78 04 ee 59 ed 33 9a e4 cc 42 bb 87 36 95 ac 4b 9b 6b 4f 96 4c ed 20 f7 bd 71 4b bc d8 91 81 3c be 56 c8 05 a6 e1 f7 66 85 88 46 91 96 a3 e4 e8 bb 45 f5 b8 78 25 4b 02 cb 23 e6 54 94 ed 94 f6 87 3d 29 55 31 eb 59 e9 a1 54 4c 50 b0 74 2c a7 b6 1e f7 23 02 03 01 00 01  
Parameters: 05 00

[Extensions]  
\* Key Usage(2.5.29.15):  
Digital Signature, Key Encipherment (a0)

Source: Fiddler Capture

| Headers  | Text View  | Syntax View | Web Forms | Hex View | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm   |
|----------|--|-------------|-----------|----------|------|---------|-----|------|-----|---|
| 00000019 | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 |             |           |          |      |         |     |      |     | com:443 HTTP/1.1..Host: www.higginbotham.com:443..Connection: keep-alive..User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36...A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below...Secure |
| 00000032 | 77 77 2E 68 69 67 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A |             |           |          |      |         |     |      |     | Protocol: TLS 1.3.Cipher Suite: TLS_AES_256_GCM_SHA384..Record Layer Version: 3.3 (TLS/1.2).Random:   |
| 0000004B | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55 |             |           |          |      |         |     |      |     | F6 9D 1E 05 3D 58 53 50 C5 38 CB 68 E9 B1 71 BE 0   |
| 00000064 | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 |             |           |          |      |         |     |      |     | 2 77 A7 FA AB 3F CC 1D 97 1B 4C AF AB 7A CD 69."Time": "21-09-1972 08:33:18. SessionID: 5E B3 4B 70 12 4D 2C CB 6A 5B 9A 63 89  |
| 0000007D | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36 |             |           |          |      |         |     |      |     |   |
| 00000096 | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48 |             |           |          |      |         |     |      |     |   |
| 000000AF | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31 |             |           |          |      |         |     |      |     |   |
| 000000CB | 32 36 2E 30 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D |             |           |          |      |         |     |      |     |   |
| 000000E1 | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E |             |           |          |      |         |     |      |     |   |
| 000000FA | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E |             |           |          |      |         |     |      |     |   |
| 00000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20 |             |           |          |      |         |     |      |     |   |
| 0000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65 |             |           |          |      |         |     |      |     |   |
| 00000145 | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72 |             |           |          |      |         |     |      |     |   |
| 0000015E | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53 |             |           |          |      |         |     |      |     |   |
| 00000177 | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69 |             |           |          |      |         |     |      |     |   |
| 00000190 | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A |             |           |          |      |         |     |      |     |   |
| 000001A9 | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20 |             |           |          |      |         |     |      |     |   |
| 000001C2 | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30 |             |           |          |      |         |     |      |     |   |
| 000001DB | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37 |             |           |          |      |         |     |      |     |   |
| 000001F4 | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69 |             |           |          |      |         |     |      |     |   |
| 0000020D | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A |             |           |          |      |         |     |      |     |   |
| 00000226 | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32 |             |           |          |      |         |     |      |     |   |
| 0000023F | 20 34 44 20 32 43 20 43 42 20 36 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |             |           |          |      |         |     |      |     |   |

Source: Fiddler Capture

|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 39 2C 7B 89 AA BB C5 AF E |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | 0C 2A 73 3E B4 85 7A 95   |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |

*Source: Fiddler Capture*

Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.

Secure Protocol: TLS 1.3  
 Cipher Suite: TLS\_AES\_256\_GCM\_SHA384

-- Server Certificate -----  
 [Version]  
 V3

[Subject]  
 CN=higginbotham.com  
 Simple Name: higginbotham.com  
 DNS Name: higginbotham.com

[Issuer]  
 CN=GTS CA 1P5, O=Google Trust Services LLC, C=US

*Source: Fiddler Capture*

The standard defines four record message types, including a handshake message type.

The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

Further, the AEAD encrypted message comprises a ciphertext (e.g., encrypted ciphertext after the encryption by the second encryption algorithm), nonce (e.g., associating second decryption algo), key and associated data. The maximum length of nonce is a cipher suit specific element. The nonce and associated data are utilized in decryption of the AEAD encrypted message.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.2. Authenticated Decryption

Second decryption algorithm

The authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic. A ciphertext C, a nonce N, and associated data A are authentic for key K when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [RFC5116].. [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:  
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

All handshake messages after the ServerHello are now encrypted.  
The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

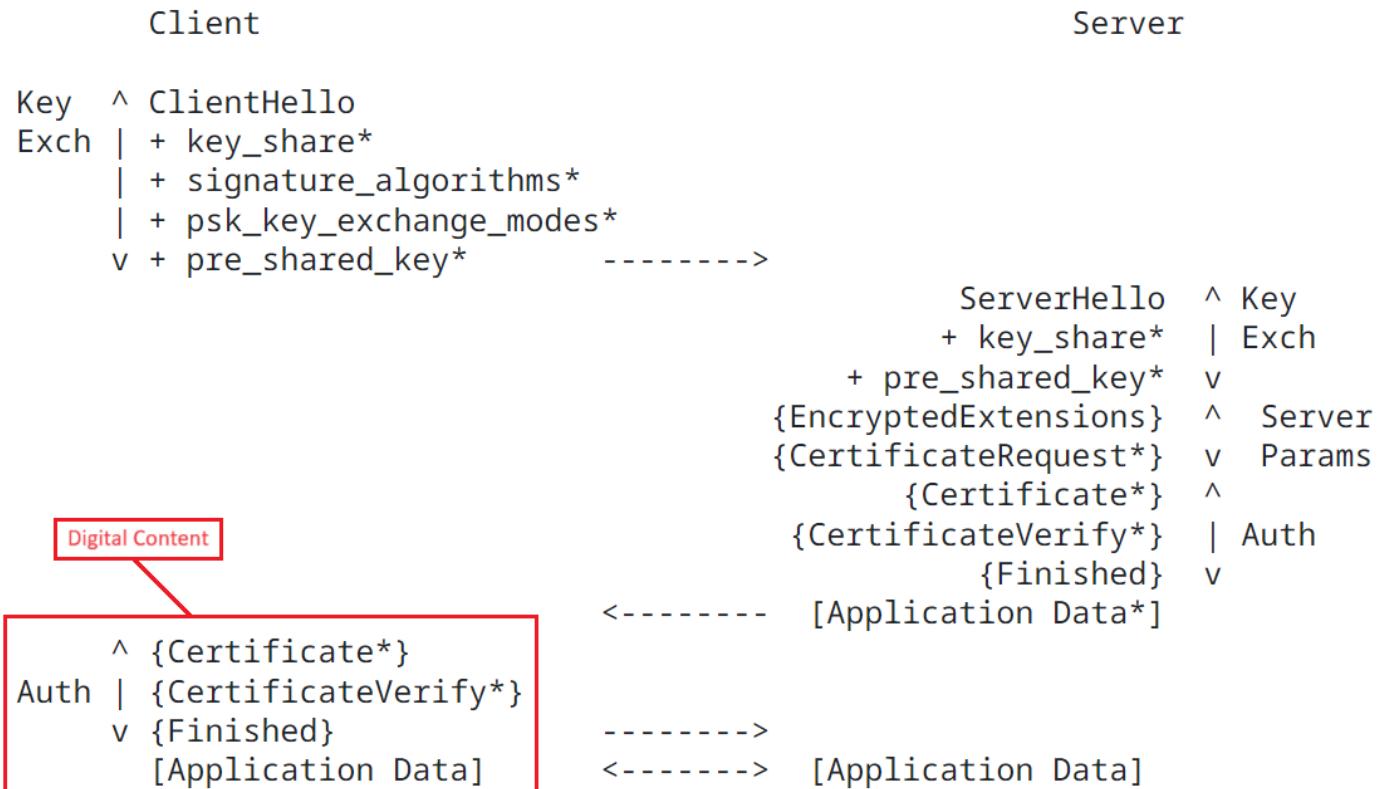
#### **4.4. Authentication Messages**

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block.

These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

As shown below, the receiving party will be able to decrypt the encrypted message with the provided signature decryption algorithm information i.e., SHA-256 RSA decryption algorithm.

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

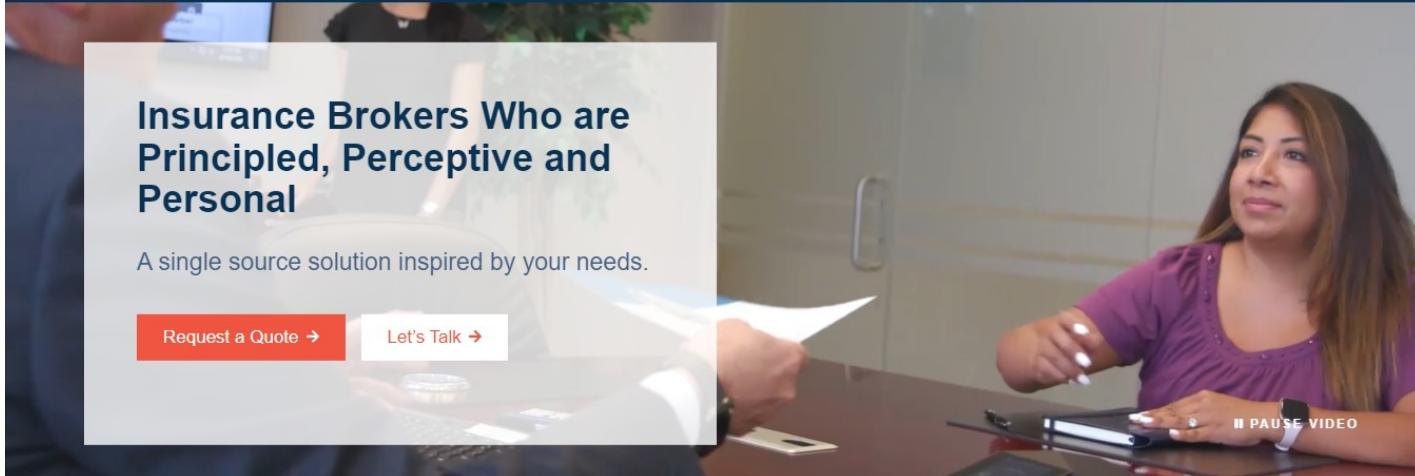
First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

|   |  |
|---|--|
|   | <p>The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A <i>cryptographic hash function</i> is a function that computes a <i>message authentication code</i> from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that <math>H</math> is a cryptographic hash function. To sign a message <math>m</math>, party A computes <math>h = D_A(H(m))</math> and sends <math>E_B(m, h)</math> to B. Party B now has evidence that A signed <math>m</math> because <math>E_A(h) = H(m)</math>, and A is the only one who could have generated a value <math>h</math> with that property.</p> |
| 3. The method of claim 2, wherein the decrypting is done using a key associated with each decryption algorithm. | The standard practices the method such that the decrypting is done using a key (e.g., decryption key) associated with each decryption algorithm (e.g., signature decryption algorithm such as SHA-256RSA, etc., and AEAD decryption algorithm such as TLS_AES_256_GCM_SHA384, etc.).   |

The Higginbotham Difference About Us Careers Blog Locations Become a Higginbotham Partner  Report a Claim

 Higginbotham\* Business Insurance Employee Benefits HR Services Financial Services Personal Insurance Captive Insurance Let's Talk



Insurance Brokers Who are Principled, Perceptive and Personal

A single source solution inspired by your needs.

[Request a Quote →](#) [Let's Talk →](#)

<https://www.higginbotham.com/>

II PAUSE VIDEO

The Google Play Store search results for "higginbotham" are displayed. The search bar shows "higginbotham". Below it, there are three categories: "Apps & games" (selected), "Movies & TV", and "Books". A search result for "Higginbotham FSA" is highlighted with a red box. The app has a rating of 4.5 stars, over 1K+ downloads, and is labeled "Everyone". A large "Install" button is present. To the right of the app's card, there are three screenshots of the app interface: "View your account(s) and link to resources from 'I Want To'", "Check your account activity anytime, anywhere", and "Manage expenses from a consolidated dashboard".

<https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US>

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP-Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 1B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB 89 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
crls_size(0x3ada) 00
```

*Source: Fiddler Capture*

As shown below, the signature decryption algorithm utilizes a private key for a first decryption and the AEAD decryption algorithm uses a key K. Both the decryption techniques are decrypting using their respective associated keys.

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

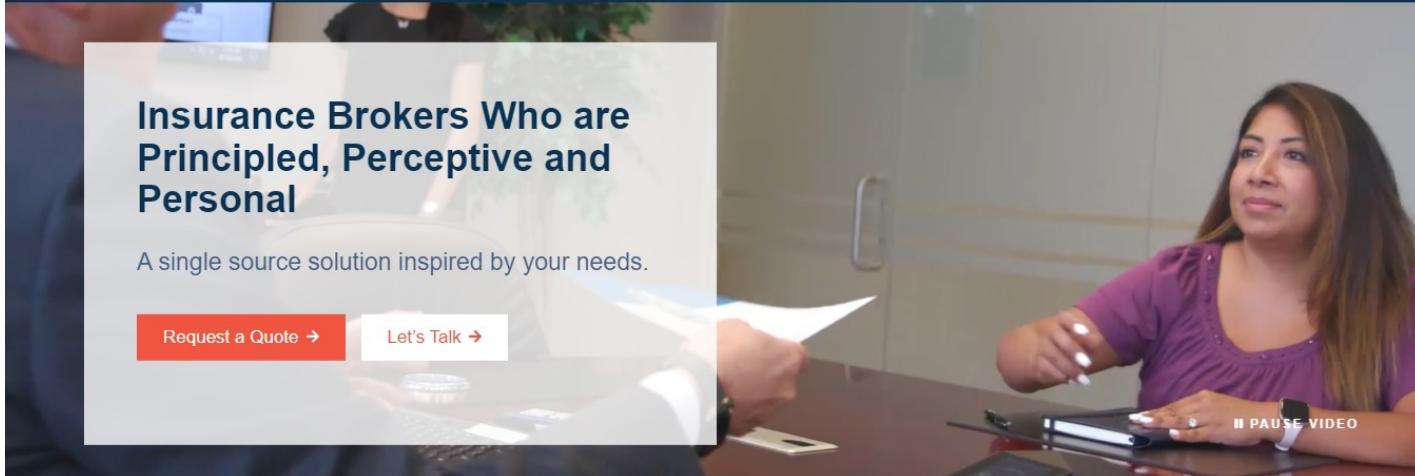
| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |  |
|---|--|
|   | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 4. The method of claim 3, wherein the key is resident in hardware of the target unit or the key is retrieved from a server. | The standard utilized by the accused instrumentality practices the method such that the key is resident in hardware (e.g., stored in a memory storage of the server such as a database, RAM, etc.) of the target unit (e.g., server of the accused instrumentality) or the key is retrieved from a server.   |

The Higginbotham Difference About Us Careers Blog Locations Become a Higginbotham Partner  Report a Claim

 Higginbotham\* Business Insurance Employee Benefits HR Services Financial Services Personal Insurance Captive Insurance Let's Talk



Insurance Brokers Who are Principled, Perceptive and Personal

A single source solution inspired by your needs.

Request a Quote → Let's Talk →

<https://www.higginbotham.com/>

II PAUSE VIDEO

The screenshot shows the Google Play Store interface. At the top, there is a search bar with the text "higginbotham". Below the search bar, there are three categories: "Apps & games" (selected), "Movies & TV", and "Books". A red box highlights the first search result, which is the "Higginbotham FSA" app. The app's icon is a blue circle with a white letter "H". It has a download count of "1K+" and is rated "Everyone". A large "Install" button is visible. To the right of the search results, there are three promotional cards for the app:

- "View your account(s) and link to resources from 'I Want To'" showing a smartphone screen with account details.
- "Check your account activity anytime, anywhere" showing a smartphone screen with account activity history.
- "Manage expenses from a consolidated dashboard" showing a smartphone screen with expense management features.

At the bottom of the page, there is a purple bar with the URL <https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US> and a "Privacy - Terms" link. Below the URL, there is a block of hex code representing a certificate or key exchange information.

**Source: Fiddler Capture**

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP-Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 58 6E
2D 8A 88 BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 FB 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
driase (0xdada) 00
```



## **Server hardware guide: Architecture, products and management**

### **3. Random access memory**



RAM is the main type of memory in a computing system.

RAM holds the software instructions and data needed by the processor, along with any output from the processor, such as data to be moved to a storage device. Thus, RAM works very closely with the processor and must match the processor's incredible speed and performance. This kind of fast memory is usually termed dynamic RAM, and several DRAM variations are available for servers.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>



## Server hardware guide: Architecture, products and management

### 4. Hard disk drive



This hardware is responsible for reading, writing and positioning of the hard disk, which is one technology for data storage on server hardware. Developed at IBM in 1953, the [hard disk drive \(HDD\)](#) has evolved over time from the size of a refrigerator to the standard 2.5-inch and 3.5-inch form factors.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>

As shown below, the server comprises a memory storage to store messages for establishing secure TLS communication. the standard discloses multiple signature encryption algorithms for a first encryption and multiple AEAD encryption algorithms for the second encryption. A signature decryption algorithm utilizes a private key for decrypting the first bitstream encrypted with the signature encryption and an AEAD decryption algorithm uses a key K for decrypting the second bitstream encrypted with the AEAD encryption. Both the decryption techniques are decrypting using their respective associated keys. A server must have a storage to store information pertaining to these algorithms and their corresponding keys such as private key, Key K, etc., to establish secure TLS communication with a client.

Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. This is necessary for the ClientHello storage mechanism described in Section 8.2 because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.

<https://datatracker.ietf.org/doc/html/rfc8446#>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

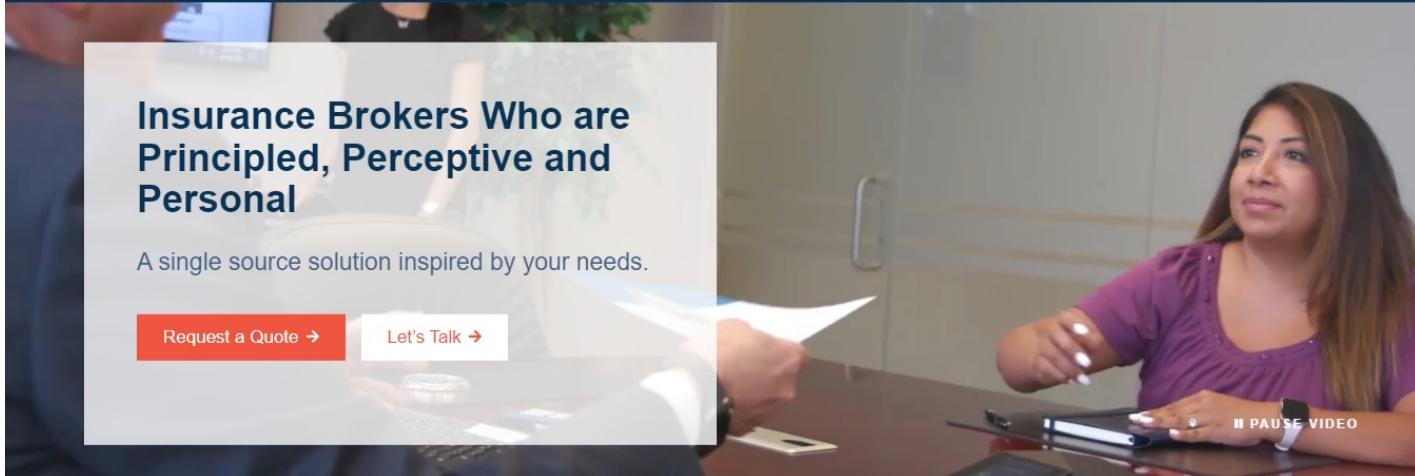
| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |  |
|---|--|
|   | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 5. The method of claim 4, wherein the key is contained in a key data structure. | The standard utilized by the accused instrumentality practices the method such that the key (e.g., private key, Key K, etc.) is contained in a key data structure (e.g., data structure).  |

The Higginbotham Difference About Us Careers Blog Locations Become a Higginbotham Partner  Report a Claim

 **Higginbotham** \* Business Insurance Employee Benefits HR Services Financial Services Personal Insurance Captive Insurance Let's Talk



Insurance Brokers Who are Principled, Perceptive and Personal

A single source solution inspired by your needs.

[Request a Quote →](#) [Let's Talk →](#)

<https://www.higginbotham.com/>

II PAUSE VIDEO

The Google Play Store search results for "higginbotham" are displayed. The top result is the "Higginbotham FSA" app, which is described as "Higginbotham Wex Health Mobile". It is categorized under "Apps & games". The app has a rating of 4.5 stars, over 1K+ downloads, and is rated "Everyone". A red box highlights the app's listing. To the right of the app listing are three screenshots of the app interface: "My Accounts", "Check your account activity", and "Manage expenses from a consolidated dashboard".

<https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US>

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP-Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 88 BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB 89 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
crls_size(0x3ada) 00
```

*Source: Fiddler Capture*

The accused instrumentality utilizes a server to establish a secure TLS communication with a client. The server must comprise a memory storage and store data according to a data structure to implement the standard efficiently.



## **Server hardware guide: Architecture, products and management**

### **3. Random access memory**



RAM is the main type of memory in a computing system.

RAM holds the software instructions and data needed by the processor, along with any output from the processor, such as data to be moved to a storage device. Thus, RAM works very closely with the processor and must match the processor's incredible speed and performance. This kind of fast memory is usually termed dynamic RAM, and several DRAM variations are available for servers.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>



## Server hardware guide: Architecture, products and management

### 4. Hard disk drive



This hardware is responsible for reading, writing and positioning of the hard disk, which is one technology for data storage on server hardware. Developed at IBM in 1953, the [hard disk drive \(HDD\)](#) has evolved over time from the size of a refrigerator to the standard 2.5-inch and 3.5-inch form factors.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>

A data structure is a specialized format for organizing, processing, retrieving and storing data.

There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose. Data structures make it easy for users to access and work with the data they need in appropriate ways. Most importantly, data structures frame the organization of information so that machines and humans can better understand it.

In computer science and computer programming, a data structure may be selected or designed to store data for the purpose of using it with various algorithms. In some cases, the algorithm's basic operations are tightly coupled to the data structure's design. Each data structure contains information about the data values, relationships between the data and -- in some cases -- functions that can be applied to the data.

<https://www.techtarget.com/searchdatamanagement/definition/data-structure>

As shown below, the server comprises a memory storage to store messages for establishing secure TLS communication. the standard discloses multiple signature encryption algorithms for a first encryption and multiple AEAD encryption algorithms for the second encryption. A signature decryption algorithm utilizes a private key for decrypting the first bitstream encrypted with the signature encryption and an AEAD decryption algorithm uses a key K for decrypting the second bitstream encrypted with the AEAD encryption. Both the decryption techniques are decrypting using their respective associated keys. A server must have a storage to store information pertaining to these algorithms and their corresponding keys such as private key, Key K, etc., to establish secure TLS communication with a client.

Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. This is necessary for the ClientHello storage mechanism described in Section 8.2 because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.

<https://datatracker.ietf.org/doc/html/rfc8446#>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |  |
|---|--|
|   | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 11. The method of claim 3, wherein each encryption algorithm is a symmetric key system or an asymmetric key system. | The standard practices the method such that each encryption algorithm (e.g., signature encryption algorithm i.e., SHA256RSA, etc., and AEAD encryption algorithm i.e., TLS_AES_256_GCM_SHA384, etc.) is a symmetric key system (e.g., AEAD encryption algorithm, etc.) or an asymmetric key system (e.g., signature encryption algorithm). <p>As shown below, the server comprises a memory storage to store messages for</p>  |

establishing secure TLS communication. the standard discloses multiple signature encryption algorithms for a first encryption and multiple AEAD encryption algorithms for the second encryption. A signature decryption algorithm utilizes a private key for decrypting the first bitstream encrypted with the signature encryption and an AEAD decryption algorithm uses a key K for decrypting the second bitstream encrypted with the AEAD encryption. The standard defines the signature encryption algorithm as an asymmetric cryptography algorithm and the AEAD encryption algorithm as the symmetric cryptography algorithm.

Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. This is necessary for the ClientHello storage mechanism described in Section 8.2 because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.

<https://datatracker.ietf.org/doc/html/rfc8446#>

Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated. Authentication can happen via asymmetric cryptography (e.g., RSA [RSA], the Elliptic Curve Digital Signature Algorithm (ECDSA) [ECDSA], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032]) or a symmetric pre-shared key (PSK).

<https://datatracker.ietf.org/doc/html/rfc8446#section-4>

cipher\_suites: A list of the symmetric cipher options supported by the client, specifically the record protection algorithm (including secret key length) and a hash to be used with HKDF, in descending order of client preference. Values are defined in [Appendix B.4](#). If the list contains cipher suites that the server does not recognize, support, or wish to use, the server MUST ignore those cipher suites and process the remaining ones as usual. If the client is attempting a PSK key establishment, it SHOULD advertise at least one cipher suite indicating a Hash associated with the PSK.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in Section 3.2, and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

|  |  |
|--|--|
|  | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 12. The method of claim 3, further comprising associating a first Message Authentication Code (MAC) or first digital signature with each | The standard practices associating a first Message Authentication Code (MAC) (e.g., message authentication code with hashing function) or first digital signature with each encrypted bit stream (e.g., encrypted bit stream with the signature encryption algorithm i.e., SHA256RSA, etc., and encrypted bitstream with the AEAD encryption algorithm i.e., TLS_AES_256_GCM_SHA384, etc.). <p>As shown below, the standard discloses a hashing function with each of the encryption</p>   |

encrypted bit stream. algorithm. It performs a message authentication code with the utilized hashing function.

```

SignedCertTimestamp (RFC6962) empty
ALPN          h2, http/1.1
signature_algs ecdsa secp256r1 sha256, rsa_pss_rsae sha256, rsa_pkcs1 sha256, ecdsa secp384r1 sha384, rsa_pss_rsae sha384, rsa_pkcs1_sha384, rsa_pss_rsae_sha512
0x001b 02 00 02
supported_groups grease [0x5a5a], unknown [0x6399], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]
key_share      04 ED 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 32 5D 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85
7A 95 44 34 07 19 C8 33 3F EC 9C 45 46 0B 84 C9 1C 22 E1 3C 9F 87 71 A3 5B C3 72 80 E1 C9 D5 CA 9D C4 69 B0 05 5B 9D E2 4B 2B 56 F5 BF 82 84 67 9C C6 65 5F 82 3E B2 F0 BB F5 3B AF 0F 86 11 A4 1C 00 5A 46 68 D9 57 AF 31 C6 C2 F4 F0 38 F1
EC 97 6A 21 78 C4 A8 AA B4 85 CE 93 85 67 D7 7A 0C EB AB 79 1D 83 3A B2 3B 60 D6 A3 C2 01 87 A7 F7 CC B8 A0 E5 1E 1F 35 88 34 65 2F BA CA 06 9C 3B 17 9D A6 11 B2 39 9F 34 00 6C DB D0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E 2F
C0 61 1D A0 54 53 73 A1 99 23 8C 39 BA A0 C6 73 7A 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A B5 2C 21 80 CA 1F E2 AC B4 73 A5 FD EB 7B 6D B5 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F E0 75 BA 39 AA 0B 66 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F
23 C1 C3 B1 5E E8 03 13 61 7C 42 DC 1C 3F F0 AB 47 A3 00 28 E6 A7 C1 C1 26 7B D7 FB 9F E9 09 14 5A E3 B6 5A 5B 9B D2 D5 47 0D 6B 7B 4A 71 6D 7E 01 37 F8 E0 A3 1E 58 6B D6 D2 C0 26 53 B8 9B B0 09 DF 5A 20 22 E9 A3 DD 00 9E 98 F0 4C A0
11 99 AD C0 48 DA D1 37 3B 2A AF 72 43 91 BD 66 6E 23 56 C7 5A E0 5E 74 33 2F 3A 86 40 07 F3 D7 95 B1 38 70 23 05 30 1B 5E A5 C7 07 13 84 84 A9 90 B3 66 B5 11 A9 A9 61 72 3C 15 DC EC 94 3E 55 93 83 64 C0 BF 79 2F C3 22 6C B1 C4 AB 5B 12
83 A0 B0 8F 26 98 2F 47 47 7B 2B 10 49 21 80 5F 14 53 0B 4E DA 90 E3 A0 82 98 C8 89 6E 63 TA 6A AA A7 E6 C8 08 2D 4B 10 BA 70 A0 3A 93 D1 75 07 59 1C 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 99 3C E8 A3 A7 F0 73 D4 FA 81 9F 99 BF 70 B7 A1
08 BB 58 A8 C2 2D C1 FB BB B7 23 89 CB F0 92 B8 A6 85 B3 E4 6C 6B 59 20 88 B5 53 12 EB 49 EB A3 13 B3 78 A5 F1 3A 70 86 8B 8B 96 96 7F 23 9C 37 19 0C C0 9C 0B 14 8F 67 78 4A F5 03 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 A4 AB
CA 6B 3C A4 85 1D 6E 70 74 88 C7 9D B9 8B 4D DD 4B 39 6D AB 00 9C 52 17 C3 C4 AD F6 10 A0 DE 0A B5 86 C7 49 SD 94 3A C5 39 95 02 BC 1E A2 73 6A 2D 16 1C 54 31 68 1D A6 C9 B5 C3 A0 12 3B 1C 72 10 60 3E 3B 23 6A 68 92 DF CC 86 91 E7 9E
2B 72 01 FC C4 34 07 61 05 82 53 94 E6 2C CC 30 74 A3 A7 08 4D 91 77 TA 14 5C 4D 4F E2 68 66 94 AD 20 34 22 4F 29 2B 29 37 42 25 08 53 R2 F4 CD A9 19 3A 44 8A 23 98 EC 22 E9 01 24 6D 29 8B DC 08 B0 45 B9 45 88 43 BA E4 99 3A 32 23 32 6C
66 31 26 FC 06 97 C2 2E 7D 39 B9 AA 17 69 3C 94 66 FA C3 3B C9 E4 5C 75 BC 59 8B 60 14 A7 26 83 FF 72 2C 0D F7 4F E9 88 B0 33 67 6D F3 C2 C0 1A 5B 69 2F 61 48 53 94 9A F6 67 00 03 7A 8C 8E BC 4E 4C 2C 49 E0 82 8C DA 88 AF DA 6A 6D B5 08
9D CA A9 A5 5A E2 91 95 3C 70 5A 3C B0 B2 AD CE 61 A9 8F C4 54 15 8B 3C A9 31 21 D8 28 CD AC 99 09 A2 37 B4 46 53 2F AF E4 AF E8 F2 55 CC FC 4F 4B 78 A4 FC 28 64 10 17 C6 61 71 2F 90 25 26 E0 C5 38 2A C6 4C A4 99 9E F8 D6 3B 1C
48 C0 AE DC B0 B6 1C 85 B3 75 4F B4 5A 14 3F C0 90 CD E6 B7 18 12 6E 34 52 C1 B5 A3 1B E4 17 72 1F F8 25 20 88 21 6B B1 A3 C3 E4 A4 2E 53 6C CC 0A 82 32 07 76 C5 23 8C C0 40 07 00 C0 B0 C2 4A 48 D3 B7 13 00 11 C5 88 73 26 F2 EA 96 15 55
CC 3E EB 0F 80 28 93 D4 49 B4 1D 52 18 44 B8 86 BD F2 BC 1E 43 34 23 C8 57 4B 65 2F E6 29 97 C8 39 39 BD E7 0F 2D C2 AF 53 DA AA 56 C1 7D 4B 93 54 D9 67 C5 D9 F5 1E 7B 86 49 57 45 85 58 64 32 42 B5 39 60 62 40 C2 96
B1 24 9A DA B4 8D FD 8A 9C SC C0 87 1E 19 12 FD FA 32 DB 69 C3 B3 T8 76 84 15 70 C9 69 73 7A F6 A7 D1 21 4C F0 63 06 42 37 TD F3 3A 85 EF D6 02 FD F1 6D 4A 3C 55 AB BA 8A 0F 45 A6 5C C8 53 19 F6 A1 75 06 90 A0 26 2D A3 90
9B AB 18 53 2C E9 0F 05 83 3D 0E 49 76 33 D8 4D 8E 72 29 2F FB 51 3B 42 5F 76 C4 AE 54 45 E6 91 8D 2B 45 F0 9A 1E 65 C0 3A D7 99 F5 AE F1 9D E9 45 D3 A3 C3 DB 7A 74 E0 3D ED 7F 6D 00 01 20 FA D2 73 2A 7D BB 08 13 72 9B 38 A0 89 F4
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP - Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 0F 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 62 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 88 3C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 FB 90 B6 F6 1A 4E 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 C6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
osrae (0x3a3a) 00

```

*Source: Fiddler Capture*

*Source: Fiddler Capture*

| Headers  | TextView   | SyntaxView  | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm |
|----------|--|---|----------|---------|------|---------|-----|------|-----|-----------------------------|
| 00000019 | 63 6F 6D 3A 34 34 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77    | com:443 HTTP/1.1..Host: www.higginbotham.com:443..Connection: keep-alive..User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/1 |          |         |      |         |     |      |     |                             |
| 00000032 | 77 77 2E 68 69 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A    | 20 78 36  |          |         |      |         |     |      |     |                             |
| 0000004B | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55 | 20 28 48  |          |         |      |         |     |      |     |                             |
| 00000064 | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 | 4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/1  |          |         |      |         |     |      |     |                             |
| 0000007D | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 28 57 | 26.0.0.0 Safari/537.36...A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below...Secure  |          |         |      |         |     |      |     |                             |
| 00000096 | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48 | Protocol: TLS 1.3.Cipher Suite: TLS_AES_256_GCM_SHA384..Record Layer Version: 3.3 (TLS/1.2).Random:   |          |         |      |         |     |      |     |                             |
| 000000AF | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31 | F6 9D 1E 05 3D 58 53 50   |          |         |      |         |     |      |     |                             |
| 000000C8 | 32 36 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D       | C5 38 CB 68 E9 B1 71 BE 0   |          |         |      |         |     |      |     |                             |
| 000000E1 | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E | 2 77 A7 FA AB 3F CC 1D 97   |          |         |      |         |     |      |     |                             |
| 000000FA | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E | 1B 4C AF AB 7A CD 69."Time": 21-09-1972 08:33:18.   |          |         |      |         |     |      |     |                             |
| 00000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20 | SessionID: ME B3 4B 70 12   |          |         |      |         |     |      |     |                             |
| 0000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65 | 4D 2C CB 6A 5B 9A 63 89   |          |         |      |         |     |      |     |                             |
| 00000145 | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72 |   |          |         |      |         |     |      |     |                             |
| 0000015E | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53 |   |          |         |      |         |     |      |     |                             |
| 00000177 | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69 |   |          |         |      |         |     |      |     |                             |
| 00000190 | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A |   |          |         |      |         |     |      |     |                             |
| 000001A9 | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20 |   |          |         |      |         |     |      |     |                             |
| 000001C2 | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30 |   |          |         |      |         |     |      |     |                             |
| 000001DB | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37 |   |          |         |      |         |     |      |     |                             |
| 000001F4 | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69 |   |          |         |      |         |     |      |     |                             |
| 0000020D | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A |   |          |         |      |         |     |      |     |                             |
| 00000226 | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32 |   |          |         |      |         |     |      |     |                             |
| 0000023F | 20 34 3A 20 32 43 20 43 42 20 36 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |   |          |         |      |         |     |      |     |                             |

|          | <i>Source: Fiddler Capture</i>   |                           |            |          |         |      |         |     |      |     |  |  |  |                  |
|----------|--|---------------------------|------------|----------|---------|------|---------|-----|------|-----|--|--|--|------------------|
|          | Headers  | TextView                  | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML |  |  |  | Second bitstream |
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 | 67 98 9F 8B 88 D9 A0 C6 1 |            |          |         |      |         |     |      |     |  |  |  |                  |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 | 5CC 37 BB 51 91 1D 30 CE  |            |          |         |      |         |     |      |     |  |  |  |                  |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 | 8D F6 8E 5F 54 C2 5A E3   |            |          |         |      |         |     |      |     |  |  |  |                  |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 | 9 2C 7B 89 AA BB C5 AF E  |            |          |         |      |         |     |      |     |  |  |  |                  |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 | 9 56 10 B8 15 57 94 A1 A1 |            |          |         |      |         |     |      |     |  |  |  |                  |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 | OC 2A 73 3E B4 85 7A 95   |            |          |         |      |         |     |      |     |  |  |  |                  |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 | 44 34 07 19 C8 33 3F EC 9 |            |          |         |      |         |     |      |     |  |  |  |                  |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 | C 45 46 0B 84 C9 1C 22 E1 |            |          |         |      |         |     |      |     |  |  |  |                  |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 | 3C 9F 87 71 A3 5B C3 72   |            |          |         |      |         |     |      |     |  |  |  |                  |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 | B0 E1 C9 D5 CA 9D C4 69 B |            |          |         |      |         |     |      |     |  |  |  |                  |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 | 0 05 5B 9D E2 A4 2B 56 F5 |            |          |         |      |         |     |      |     |  |  |  |                  |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 | BF 82 84 67 9C C6 65 5F   |            |          |         |      |         |     |      |     |  |  |  |                  |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 | 82 3E B2 F0 BB F5 3B AF 0 |            |          |         |      |         |     |      |     |  |  |  |                  |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 | F 86 11 A4 1C 00 5A 46 69 |            |          |         |      |         |     |      |     |  |  |  |                  |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 | D9 57 AF 31 C6 C2 F4 F0   |            |          |         |      |         |     |      |     |  |  |  |                  |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 | 38 F7 EC 97 6A 21 78 C4 A |            |          |         |      |         |     |      |     |  |  |  |                  |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 | 8 AA B4 85 CE C9 85 67 D7 |            |          |         |      |         |     |      |     |  |  |  |                  |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 | 7A 0C EB AB 79 1D 83 3A   |            |          |         |      |         |     |      |     |  |  |  |                  |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 | B2 3B 60 D6 A3 C2 01 87 A |            |          |         |      |         |     |      |     |  |  |  |                  |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 | 7 F7 CC B8 A0 E5 1E 1F 55 |            |          |         |      |         |     |      |     |  |  |  |                  |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 | 88 34 65 2F BA CA 06 9C   |            |          |         |      |         |     |      |     |  |  |  |                  |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 | 3B 17 9D A6 11 B2 39 9F 3 |            |          |         |      |         |     |      |     |  |  |  |                  |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 | 4 00 6C CB D0 53 3E 57 16 |            |          |         |      |         |     |      |     |  |  |  |                  |

*Source: Fiddler Capture*

|                                       |   |
|---------------------------------------|---|
|                                       | <p>The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A <i>cryptographic hash function</i> is a function that computes a <i>message authentication code</i> from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that <math>H</math> is a cryptographic hash function. To sign a message <math>m</math>, party <math>A</math> computes <math>h = D_A(H(m))</math> and sends <math>E_B(m, h)</math> to <math>B</math>. Party <math>B</math> now has evidence that <math>A</math> signed <math>m</math> because <math>E_A(h) = H(m)</math>, and <math>A</math> is the only one who could have generated a value <math>h</math> with that property.</p> <p><a href="https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf">https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf</a></p> <p>The list of supported symmetric encryption algorithms has been pruned of all algorithms that are considered legacy. Those that remain are all Authenticated Encryption with Associated Data (AEAD) algorithms. The cipher suite concept has been changed to separate the authentication and key exchange mechanisms from the record protection algorithm (including secret key length) and a <u>hash to be used with both the key derivation function and handshake message authentication code (MAC)</u>.</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-4">https://datatracker.ietf.org/doc/html/rfc8446#section-4</a></p> |
| 19. A system for a recursive security | The accused instrumentality utilizes a system for a recursive security protocol (e.g., TLS 1.3 security protocol) for protecting digital content (e.g., digital certificate related   |

protocol for protecting digital content, comprising a processor to execute instructions and a memory operable to store instructions for performing the steps of:

to the accused instrumentality), comprising a processor (e.g., a processor of the server of the accused instrumentality) to execute instructions and a memory (e.g., a memory of the server of the accused instrumentality) operable to store instructions.

The accused instrumentality utilizes TLS 1.3 security protocol (hereinafter “the standard”) for communicating content such as digital certificate, application data, etc., with a client. The standard provides a two-level encryption security. It encrypts a plaintext with a first encryption technique and generates a ciphertext. Further, it encrypts the ciphertext with a second encryption technique i.e., recursive encryption security.



<https://www.higginbotham.com/>

The screenshot shows the Google Play Store interface. At the top, there is a search bar with the text "higginbotham". Below the search bar, there are three categories: "Apps & games" (selected), "Movies & TV", and "Books". To the right of the search bar are a magnifying glass icon, an "X" button, and a user profile icon.

The main search results are displayed below. The first result is highlighted with a red box and labeled "Higginbotham FSA Higginbotham Wex Health Mobile". It features a large image of a smartphone displaying the app's interface, which includes sections for "My Accounts", "I Want To", and "Account Activity". Below this, there is a summary of account details: FSA (\$2,400.00), HSA (\$1,900.00), and DependentCare (\$2,300.00). The app has over 1K+ downloads and is rated E for Everyone. A large "Install" button is present at the bottom of this card.

Below the main result, there are two more cards showing smartphone screenshots of the app's interface, each with a "View Details" button below it. The first screenshot shows the "Account Activity" screen with a list of transactions. The second screenshot shows the "Dashboard" screen with a "CREATE NEW EXPENSE" button and a list of expenses.

At the bottom of the page, there is a purple horizontal bar containing the URL <https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US>. To the right of the URL is a small blue square icon with a white letter "C" and the text "Privacy - Terms".

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

The Transport Layer Security (TLS) Protocol Version 1.3

## Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

As shown below, the accused instrumentality utilizes a two-level algorithm security. It utilizes the SHA256RSA encryption algorithm as a first encryption algorithm i.e., signature encryption algorithm and the TLS\_AES\_256\_GCM\_SHA384 encryption algorithm as a second encryption algorithm i.e., AEAD encryption algorithm.

| 8  | 200 | HTTP  | Tunnel to www.higginbotham.com:443                   | 3,612  |  |
|----|-----|-------|--|--------|--|
| 9  | 200 | HTTPS | www.higginbotham.com /                               | 17,353 |  |
| 10 | 200 | HTTPS | www.googleapis.com /oauth2/v4/token                  | 399    |  |
| 11 | 200 | HTTPS | optimizationguide-p... /v1:Get-Links?                | 428    |  |
| 12 | 200 | HTTP  | Tunnel to safebrowsing.google.com...                 | 10,052 |  |
| 13 | 200 | HTTPS | safebrowsing.google.com /safebrowsing/clientrepor... | 32     |  |
| 14 | 200 | HTTPS | www.higginbotham.com /wp-includes/css/dist/bloc...   | 14,716 |  |
| 15 | 200 | HTTPS | www.higginbotham.com /wp-content/cache/min/1/...     | 35,238 |  |
| 16 | 200 | HTTP  | Tunnel to cdnjs.cloudflare.com:443                   | 2,748  |  |
| 17 | 200 | HTTP  | Tunnel to cdn.callrail.com:443                       | 4,277  |  |
| 18 | 200 | HTTPS | www.higginbotham.com /wp-content/cache/min/1/...     | 6,980  |  |
| 19 | 200 | HTTPS | www.higginbotham.com /wp-content/cache/min/1/...     | 23,260 |  |
| 20 | 200 | HTTPS | www.higginbotham.com /wp-content/cache/min/1/...     | 23,266 |  |
| 21 | 200 | HTTPS | www.higginbotham.com /wp-content/themes/orbit...     | 26,907 |  |
| 22 | 200 | HTTPS | www.higginbotham.com /wp-content/cache/min/1/...     | 1,436  |  |
| 23 | 200 | HTTPS | www.higginbotham.com /wp-includes/js/jquery/jqu...   | 31,396 |  |

*Source: Fiddler Capture*

SignedCertTimestamp(RFC6962) empty  
ALPN h2, http/1.1  
signature\_algs ecdsa\_secp256r1\_sha256\_rsa\_pss\_rsae\_sha256\_rsa\_pkcs1\_sha256\_ecdsa\_secp384r1\_sha384\_rsa\_pss\_rsae\_sha384\_rsa\_pkcs1\_sha384\_rsa\_pss\_rsae\_sha512\_rsa\_pkcs1\_sha512  
0x001b 02 00 02  
supported\_groups grease [0x5a5], unknown [0x6399], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]  
key\_share 04 ED 5A 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 32 5D 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8F 6E 8F 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85  
7A 95 44 3C 83 33 C8 9C 45 22 E1 3C 9F 87 71 A3 5B C3 72 BD E1 C9 5D CA 9D C4 69 80 05 5B 9D E2 A4 2B 56 F5 FB 84 67 9C C6 65 5F 82 3E B2 F5 2B AF 06 11 A4 1C 00 5A 46 69 D9 57 AF 31 C6 C2 F4 F0 38 F1  
EC 97 6A 21 78 C4 A8 AA B4 85 C9 85 67 D7 7A 0C EB AB 79 1D 83 A3 B2 3B 60 D6 A3 C2 01 87 AT F7 CC B8 A0 E5 1E 1F 55 88 34 65 2F BA CA 06 9C 3B 17 9D A6 11 B2 39 9F 34 00 6C BD D0 53 3E 57 16 06 E2 72 B7 D8 42 3E EA 03 9B DB 8E 2F  
C6 01 7A C1 54 88 55 66 2D 3C 1B 3A 59 88 2B 3E 8A 00 C6 73 A4 59 48 6B 1A 69 74 99 45 55 5B BC DA 5B 2C 21 80 CA 1F E2 AC B4 73 A5 FD E2 7B 6D 5B 93 10 37 AT A3 C8 18 C6 49 D3 71 0F 0E 75 BA 39 AA 08 66 56 6D DC 3C 1E 5D 8A 7E 1A B6 2E F3  
23 C1 C3 B1 5E E8 05 13 61 7C 42 DC 1C 3F P0 FA 47 A3 00 28 E6 A7 C1 C1 26 7B D7 FB 9F E9 09 14 5A E3 B6 5A 5B 9B D2 D5 47 0D 6B 7B 8A 71 6D TE 01 37 F8 E0 A3 1E 58 6B D6 D2 C0 26 53 B8 98 80 09 DF 5A 20 22 E9 A3 DD D0 9E 98 F0 4C A0  
11 99 AD C0 48 DA D1 37 3B 2A AF 72 43 91 BD 66 2E 23 56 C7 5A E0 5E 74 33 BF 3A 86 40 07 F7 D3 95 B1 38 70 23 05 30 1B 5E A5 C7 07 13 84 A9 90 B3 66 B5 11 A9 A9 61 72 3C 15 DC EC 94 3E 55 93 83 64 C0 8F 79 2F C3 22 6C B1 C4 A8 5B 12  
83 A0 B6 26 98 2F 47 47 B2 10 49 21 80 5F 14 33 0B 4E 03 A2 88 2E 63 9A 88 44 7A 8A AA 07 0E 2D 4B 70 1D 82 96 C3 E8 A3 A7 F0 73 D4 10 49 81 7F 1D 8B 92 2E 8A 03 A7 F0 73 D4 8A 81 9F 99 BF 77 A1  
08 BB 58 A8 CC 2D 1C FB BB 73 89 CB F0 92 BB A8 65 B3 E4 6C 66 59 20 88 B5 53 12 EB 49 EB A3 13 B7 78 A5 F1 3A 70 86 88 86 96 7F 23 9C 37 37 19 0C C0 3C 08 14 8F 67 78 4A F5 03 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 16 15 A4 AB  
CA 6B 3C 85 2C 1D FB BB 73 89 CB F0 92 BB A8 65 B3 E4 6C 66 59 20 88 B5 53 12 EB 49 EB A3 13 B7 78 A5 F1 3A 70 86 88 86 96 7F 23 9C 37 37 19 0C C0 3C 08 14 8F 67 78 4A F5 03 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 16 15 A4 AB  
2B 72 01 7B C4 34 07 61 05 82 53 94 E6 2C CC 30 74 TA A7 08 04 1K 91 77 TA 14 5C 4D F4 E2 68 66 94 AD 20 34 22 4F 29 2B 29 37 42 25 53 92 FD CA D9 13 8A 44 2A 23 98 EC 22 E9 01 24 6D 29 B8 DC 0B 80 45 B9 45 88 43 BA E4 99 3A 32 23 32 8C 66  
31 26 FC 06 97 C2 2E 7D 39 B9 AA 17 69 94 66 FA C3 B8 C9 E4 5C 75 BC 59 88 60 14 A7 26 83 FF 72 2C 0D F7 F4 E8 88 80 33 67 6D F3 C2 C0 1A 5B 69 2F 61 48 53 94 9A FC 67 00 03 7A 8C 8E CB 4E 4C 2C 49 E0 82 8C DA 8B AF DA 6A 6D 85 08  
9D CA 9A A5 5A E2 91 95 7C 30 5C 3C B0 D2 AD C6 E1 A9 F8 C4 54 2B 18 28 CD AC 99 09 A2 37 B4 46 53 2F AF E4 AF E8 F2 55 CC FC 4F 4B 7A 4C FC 28 64 10 17 C6 61 71 2F 90 25 26 E0 C5 38 2A C6 4C A4 99 9E F8 D8 3B 48  
40 C0 AE DC B0 86 1C 85 B3 75 4F BA 54 14 3F C0 90 CD E6 87 18 12 6E 34 52 C1 B5 A3 1B E4 17 72 1F F8 25 20 88 21 6B B1 A3 1C E4 A2 53 6C C0 0A 82 32 07 76 C5 23 8C C0 40 07 00 C0 B0 C2 4A 48 D3 B7 13 00 11 C5 88 73 26 F2 EA 96 15 55  
CC 3E EB OF 08 28 93 D4 49 B1 52 18 88 66 FD 5C B8 AD 86 B0 FF B2 F2 C1 E4 33 42 C3 28 57 B4 65 2F E7 07 76 C5 39 9B D7 07 76 C5 39 9B D7 05 53 AA 56 C1 7D 48 93 54 9D 67 C5 9F 1E 7B 49 57 45 86 24 32 8C 59 60 62 40 C2 96  
B1 24 9A DA 8B FD 8A 8C 5C D0 87 1E 19 12 FD 32 DB 69 C3 B7 78 64 15 70 C6 79 73 A6 7A 66 D7 A1 21 4C F0 63 06 42 37 TD F3 3A 85 D0 85 E6 F6 02 FD 1F 6D 4A 3C 55 5C AB BA 8A 0F 45 A6 5C C8 53 19 F6 A1 75 96 80 26 2D A3 90  
9B AB 18 53 2C E9 OF 05 83 3D 0E 49 76 33 D8 4D 8E 72 29 2F FB 51 3B 42 5F 76 C4 AE 54 45 E6 91 8D 2B 45 F0 9A E1 65 C0 3A D7 99 F5 AE F1 9D 45 D3 A3 CD DB 7A 74 E0 3D ED 7F 6D 00 1D 00 20 FA D2 73 2A 7D BB 08 13 72 9B 38 A0 89 F4  
7A 36 23 01 54 D1 F6 FB F9 09 82 32 D7 CC C6 34 38 24  
extended\_master\_secret empty  
ec\_point\_formats uncompressed [0x0]  
status\_request OCSP - Implicit Responder  
psk\_key\_exchange\_modes 01 01  
renegotiation\_info 00  
0xfe0d 00 00 01 00 01 00 00 20 08 0F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E1 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15  
F6 5E AC B2 14 57 6C 00 FB C9 30 9C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E  
2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB B9 04 32 E5 A7 B5 38 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 F8 90 B6 B6 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 C  
E4 C8 4A D3 F3 3C D8 8F 8A DF C3 0A 9F 27 E4 C6 05 53 F8 33 8B 95 A

*Source: Fiddler Capture*

SignedCertTimestamp (RFC6962) empty  
ALPN h2, http/1.1  
signature\_algs ecdsa\_secp256r1\_sha256\_rsa\_pss\_rsae\_sha256\_rsa\_pkcs1\_sha256\_ecdsa\_secp384r1\_sha384\_rsa\_pss\_rsae\_sha384\_rsa\_pkcs1\_sha384\_rsa\_pss\_rsae\_sha512\_rsa\_pkcs1\_sha512  
0x001b 02 00 02  
supported\_groups grease [0x5a5a], unknown [0x6399], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]  
First encryption algorithm  
key\_share 04 ED 5A 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 32 5D 09 69 67 98 9F 8B 89 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85  
7A 95 44 32 93 33 3F C9 1C 95 46 08 84 C9 1C 93 3C 9F 87 71 A3 5B C3 72 B2 E1 C9 D5 CA 9D C4 60 80 5B 97 E2 A4 2B 96 F5 B2 84 67 9C C6 65 5F 82 3E B2 FF B5 3B AF D0 86 11 A4 0C 00 5A 46 69 D9 57 AF 31 C6 C2 F4 F0 38 F1  
EC 97 6A 21 78 C4 A8 AB 84 85 C9 0C EB 77 0D 77 A0 7D 77 0C EB 79 1D 83 DA 2B 3B 60 D6 A3 C2 01 87 A7 F7 CC B8 A0 E5 1E 01 55 88 65 2F CA 06 C9 3B 17 6D A1 66 B2 39 9F 34 00 6C DB 03 53 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E 2F  
C0 61 D0 54 55 73 A1 99 23 8C 39 A0 C6 73 9A 54 95 8A 6B 1A 69 74 99 55 55 BC BA 05 B2 21 80 CA 1F E2 AC B4 73 A5 FD E8 7B 6D B5 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F E0 75 BA 39 AA 08 66 56 6D DC 3C 1E 5D 8A 7E 1A B6 2E F1  
23 C1 C3 B1 E5 E8 05 13 61 7C 42 DC 1C 3F P0 FA 47 A3 00 28 E6 A7 C1 C6 2B D7 FB 9F E9 09 14 5A E3 B6 5A 5B 9B D2 D5 47 0D 6B 7B 8A 4A 71 6D 7E 01 37 F9 E0 A3 1E 58 6B D6 D2 C0 26 53 B8 98 80 09 DF 5A 20 22 E9 A3 DD D0 9E 98 F0 4C A0  
11 99 AD C0 48 DA D1 37 3B 2A AF 72 43 91 BD 66 E6 23 56 C7 5A E0 5E 74 33 BF 3A 86 40 07 F3 7D 85 B1 38 70 23 05 30 1B 5E A5 C7 07 13 84 84 A9 90 B3 66 B5 11 A9 A9 61 72 3C 15 DC EC 94 3E 55 93 83 64 C0 B7 79 2F C3 22 6C B1 C4 AB 5B 12  
83 A0 B0 86 26 98 2B 47 47 7B 28 10 49 80 5F 14 53 0B 84 DE 03 A8 92 B8 2A 73 8A AA 07 E8 2D 4B 10 BA 70 A0 3A 93 D1 75 07 39 8A 90 B3 66 B5 11 A9 A9 61 72 3C 15 DC EC 94 3E 55 93 83 64 C0 B7 79 2F C3 22 6C B1 C4 AB 5B 12  
08 BB 58 A8 CC 2D 1C FB BB 73 89 CB F0 92 BB AB 65 B3 E4 6C 68 59 20 88 B5 53 12 EB 49 EB A3 13 B3 78 A5 F1 A3 70 86 88 96 96 7F 23 39 19 0C C0 3C 08 14 1F 6B 77 84 PA 03 04 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 AA 4B  
CA 6B 3C A4 61 86 27 C9 7D 9B 88 4D D4 3B 9D AB 00 9C 52 17 C3 C3 AD F6 10 A0 DE 04 B5 SE 47 9D 94 3A C9 35 92 05 2C 1E 73 6A 2D 16 51 31 68 1D A9 C9 B5 C3 A0 12 3B 1C 72 1D 60 3E 7B 23 6A 68 92 DF CC 86 91 E7 98  
2B 72 01 7B C4 34 07 61 05 82 53 94 E6 2C CC 30 74 TA A7 08 04 91 77 TA 14 5C 4D F2 E6 88 96 94 AD 20 34 22 4F 29 2B 29 37 42 25 03 92 FD CA D9 13 9A 44 8A 23 98 EC 22 E9 01 24 6D 29 B8 DC 08 B0 45 B9 45 88 43 EA 94 99 3A 32 23 32 G6  
31 26 FC 06 97 C2 E6 7D 39 B9 AA 17 69 3C 94 66 FA C3 C3 B9 E4 5C 75 BC 59 8B 60 14 A7 26 83 FF 72 C2 0D F7 F4 E9 88 B0 33 67 6D F3 C2 C0 1A 5B 69 2F 61 48 33 94 9A FC 67 00 03 7A 8C 8E CB 4E 4C 2C 49 E0 82 8C DA B8 AF DA 6A 6D B5 08  
9D CA 5A E2 91 A8 9F C4 54 15 8B 3C A9 31 21 D8 2B CD AC 99 09 A2 37 B4 46 53 2F AF E4 AF E8 25 55 CC FC 4F 4B 7A 84 FC 28 64 10 17 C6 61 71 2F 90 25 26 E0 C3 38 2A C6 4A 99 F8 D8 3B 1C  
48 CO AE DC B0 86 1C 85 83 75 4F BA 54 14 3F C0 90 CD E6 B7 18 12 6E 34 52 C1 B5 A3 B1 E4 17 72 1F F8 25 20 88 21 6B 1A 3C E4 A4 2E 53 6C CC 0A 82 32 07 76 C5 23 8C C0 40 07 00 C0 B0 C2 4A 48 D3 B7 13 00 11 C5 88 73 26 F2 EA 96 15 55  
CC 84 DA B4 8D FD 8A 9C 5C C0 87 1E 19 12 FD FA 32 D8 69 C3 B3 7B 76 84 15 70 C9 69 73 TA F6 A6 D7 A1 21 4C F0 63 06 42 37 TD F3 A3 B5 D0 85 EF D6 02 FD F1 6D 4A 3C 55 AB BA 8A 0F 45 A6 5C C8 53 19 F6 A1 75 06 90 A2 26 2D A3 90  
9B AB 18 53 2C E9 0F 05 83 3D 0E 49 76 33 D8 4D 8E 72 29 2F FB 51 3B 42 5F 76 C4 AE 54 45 E6 91 8D 2B 45 F9 A1 6E 05 C0 3A D7 99 F5 AEF1 9D E9 45 D3 A3 CD DB 7A 74 E0 3D ED 7F 6D 00 1D 00 20 FA D2 73 2A 7D BB 08 13 72 9B 38 A0 89 F4  
7A 36 23 01 54 1D F1 F6 FB F9 09 92 32 D7 C6 34 38 24  
extended\_master\_secret empty  
ec\_point\_formats uncompressed [0x0]  
status\_request OCSP-Implicit Responder  
psk\_key\_exchange\_modes 01 01  
renegotiation\_info 00  
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 E4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15  
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E B5 33 09 60 EB EC 99 5C D3 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B 86 B6 52 TA 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E  
2D 8A 8B FD 95 0F D4 63 E1 FC 35 82 4F 90 98 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 FB 90 F6 B6 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 C  
E4 C8 4A D5 F3 C3 D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A  
server\_name www.higginbotham.com  
dssae [0x3a3a] 00  
Source: Fiddler Capture

[Thumbprint]  
129157F7A5852AD23DF17FE2934E6396DB6B0251

[Signature Algorithm]  
sha256RSA(1.2.840.113549.1.1.11)

First decryption  
algorithm

[Public Key]  
Algorithm: RSA  
Length: 2048

Key Blob: 30 82 01 0a 02 82 01 01 00 ca b8 66 5f 8a 37 94 14 e1 2f e3 49 cf d7 ad cb 20 8c 54 93 6b af 66 56 02 11 26 db 20 13 00 a0 ce fe ae e9 ac 8d 68 8f a0 56 fd ff b4 14 b9 c5 54 ff 55 4a 95 76 b9 db 3c 77 5e 9a b1 dd 30 74 36 77 cb 92 12 df 17 62 6d 2a aa 74 3c f6 68 c8 34 8d eb 69 97 9e f2 43 02 fa d7 76 99 c0 bc b3 56 2d 17 ca 39 c8 00 5d 91 c8 1d 9b c9 70 c8 c1 b0 da 40 23 3e 9a fe a9 ed e4 60 ce 15 b0 af 43 a3 07 a5 81 25 dd c3 92 e7 72 9e 27 df bd 01 05 2a b4 60 b8 83 6b fa bc db 97 51 47 a8 2e d0 52 7f c2 0a 1c 02 96 d2 c7 fd 04 a5 7e 78 04 ee 59 ed 33 9a e4 cc 42 bb 87 36 95 ac 4b 9b 6b 4f 96 4c ed 20 f7 bd 71 4b bc d8 91 81 3c be 56 c8 05 a6 e1 7f 66 85 88 46 91 96 a3 e4 e8 bb 45 f5 b8 78 25 4b 02 cb 23 e6 54 94 ed 94 f6 87 3d 29 55 31 eb 59 e9 a1 54 4c 50 b0 74 2c a7 b6 1e 7f 23 02 03 01 00 01  
Parameters: 05 00

[Extensions]  
\* Key Usage(2.5.29.15):  
Digital Signature, Key Encipherment (a0)

*Source: Fiddler Capture*

| Headers  | Text View  | Syntax View | Web Forms | Hex View | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm   |
|----------|--|-------------|-----------|----------|------|---------|-----|------|-----|---|
| 00000019 | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 |             |           |          |      |         |     |      |     | com:443 HTTP/1.1..Host: www.higginbotham.com:443..Connection: keep-alive..User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36...A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below...Secure |
| 00000032 | 77 77 2E 68 69 67 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A |             |           |          |      |         |     |      |     | Protocol: TLS 1.3.Cipher Suite: TLS_AES_256_GCM_SHA384..Record Layer Version: 3.3 (TLS/1.2).Random:   |
| 0000004B | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55 |             |           |          |      |         |     |      |     | F6 9D 1E 05 3D 58 53 50 C5 38 CB 68 E9 B1 71 BE 0   |
| 00000064 | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 |             |           |          |      |         |     |      |     | 2 77 A7 FA AB 3F CC 1D 97 1B 4C AF AB 7A CD 69."Time": "21-09-1972 08:33:18. SessionID: 5E B3 4B 70 12 4D 2C CB 6A 5B 9A 63 89  |
| 0000007D | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36 |             |           |          |      |         |     |      |     |   |
| 00000096 | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48 |             |           |          |      |         |     |      |     |   |
| 000000AF | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31 |             |           |          |      |         |     |      |     |   |
| 000000CB | 32 36 2E 30 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D |             |           |          |      |         |     |      |     |   |
| 000000E1 | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E |             |           |          |      |         |     |      |     |   |
| 000000FA | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E |             |           |          |      |         |     |      |     |   |
| 00000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20 |             |           |          |      |         |     |      |     |   |
| 0000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65 |             |           |          |      |         |     |      |     |   |
| 00000145 | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72 |             |           |          |      |         |     |      |     |   |
| 0000015E | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53 |             |           |          |      |         |     |      |     |   |
| 00000177 | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69 |             |           |          |      |         |     |      |     |   |
| 00000190 | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A |             |           |          |      |         |     |      |     |   |
| 000001A9 | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20 |             |           |          |      |         |     |      |     |   |
| 000001C2 | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30 |             |           |          |      |         |     |      |     |   |
| 000001DB | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37 |             |           |          |      |         |     |      |     |   |
| 000001F4 | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69 |             |           |          |      |         |     |      |     |   |
| 0000020D | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A |             |           |          |      |         |     |      |     |   |
| 00000226 | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32 |             |           |          |      |         |     |      |     |   |
| 0000023F | 20 34 44 20 32 43 20 43 42 20 36 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |             |           |          |      |         |     |      |     |   |

*Source: Fiddler Capture*

|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 39 2C 7B 89 AA BB C5 AF E |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | 0C 2A 73 3E B4 85 7A 95   |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |

*Source: Fiddler Capture*

Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.

Secure Protocol: TLS 1.3  
 Cipher Suite: TLS\_AES\_256\_GCM\_SHA384

-- Server Certificate -----  
 [Version]  
 V3

[Subject]  
 CN=higginbotham.com  
 Simple Name: higginbotham.com  
 DNS Name: higginbotham.com

[Issuer]  
 CN=GTS CA 1P5, O=Google Trust Services LLC, C=US

*Source: Fiddler Capture*

As shown below, the server of the accused instrumentality comprises a processor to

execute instructions and a memory storage to store instructions for performing the operations defined by the standard.

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP - Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 88 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 B6 FB 61 AE 44 A1 D9 62 78 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
orase (0x3a3a) 00
```

*Source: Fiddler Capture*



## Server hardware guide: Architecture, products and management



### 2. Processor

The CPU -- or simply processor -- is a complex micro-circuitry device that serves as the foundation of all computer operations. It supports hundreds of possible commands hardwired into hundreds of millions of transistors to process low-level software instructions -- microcode -- and data and derive a desired logical or mathematical result. The processor works closely with memory, which both holds the software instructions and data to be processed as well as the results or output of those processor operations.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>



## **Server hardware guide: Architecture, products and management**

### **3. Random access memory**



RAM is the main type of memory in a computing system.

RAM holds the software instructions and data needed by the processor, along with any output from the processor, such as data to be moved to a storage device. Thus, RAM works very closely with the processor and must match the processor's incredible speed and performance. This kind of fast memory is usually termed dynamic RAM, and several DRAM variations are available for servers.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>



## Server hardware guide: Architecture, products and management

### 4. Hard disk drive



This hardware is responsible for reading, writing and positioning of the hard disk, which is one technology for data storage on server hardware. Developed at IBM in 1953, the [hard disk drive \(HDD\)](#) has evolved over time from the size of a refrigerator to the standard 2.5-inch and 3.5-inch form factors.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>

Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. This is necessary for the ClientHello storage mechanism described in [Section 8.2](#) because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.

<https://datatracker.ietf.org/doc/html/rfc8446#>

The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

All handshake messages after the ServerHello are now encrypted. The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

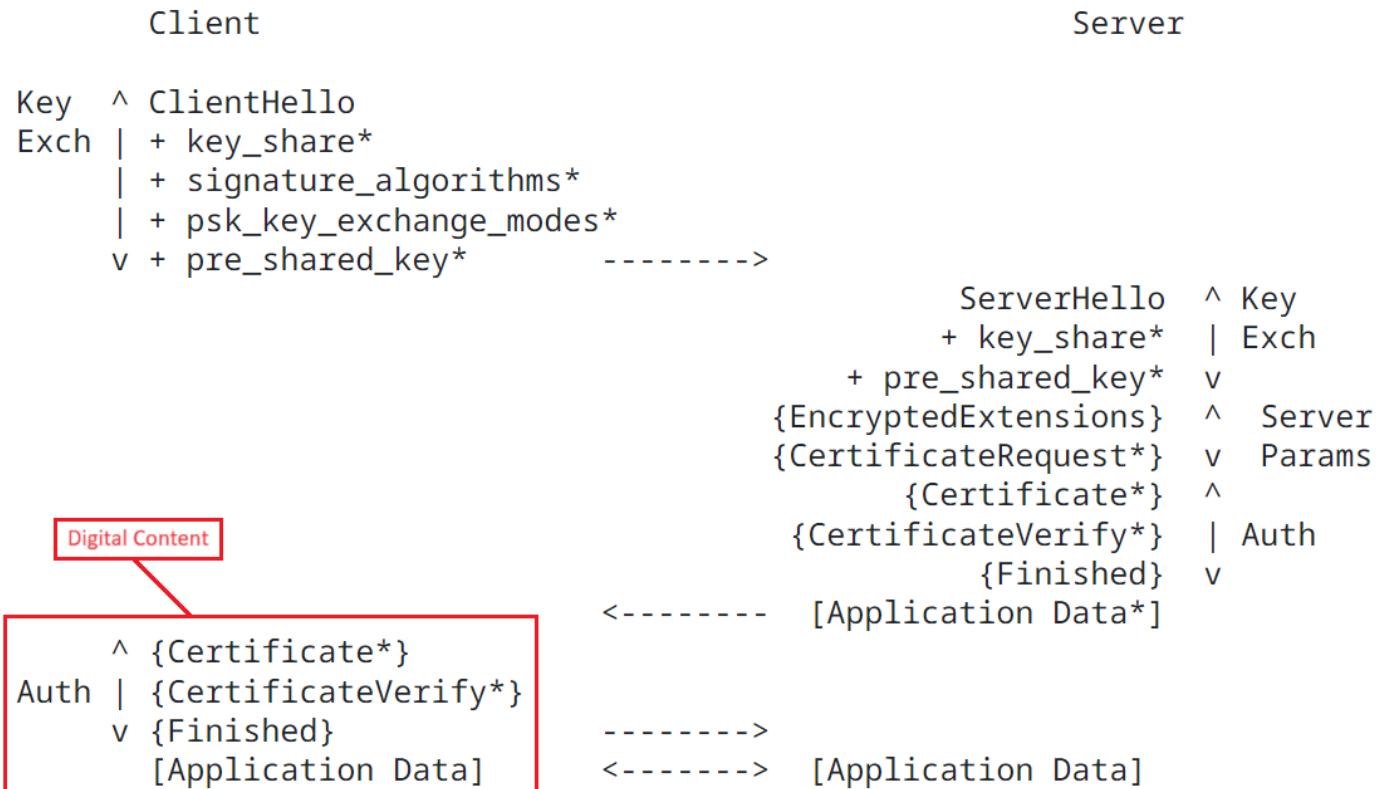
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]](#)\_handshake\_traffic\_secret.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## Introduction

The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream. Specifically, the secure channel should provide the following properties:

- Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated.  
Authentication can happen via asymmetric cryptography (e.g., RSA [[RSA](#)], the Elliptic Curve Digital Signature Algorithm (ECDSA) [[ECDSA](#)], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [[RFC8032](#)]) or a symmetric pre-shared key (PSK).
- Confidentiality: Data sent over the channel after establishment is only visible to the endpoints. TLS does not hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.
- Integrity: Data sent over the channel after establishment cannot be modified by attackers without detection.

First encryption

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [\[RFC5116\]](#), [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|  | <p>This specification defines the following cipher suites for use with TLS 1.3.</p> <table border="1"><thead><tr><th>Description</th><th>Value</th></tr></thead><tbody><tr><td>TLS_AES_128_GCM_SHA256</td><td>{0x13,0x01}</td></tr><tr><td>TLS_AES_256_GCM_SHA384</td><td>{0x13,0x02}</td></tr><tr><td>TLS_CHACHA20_POLY1305_SHA256</td><td>{0x13,0x03}</td></tr><tr><td>TLS_AES_128_CCM_SHA256</td><td>{0x13,0x04}</td></tr><tr><td>TLS_AES_128_CCM_8_SHA256</td><td>{0x13,0x05}</td></tr></tbody></table> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> | Description | Value | TLS_AES_128_GCM_SHA256 | {0x13,0x01} | TLS_AES_256_GCM_SHA384 | {0x13,0x02} | TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} | TLS_AES_128_CCM_SHA256 | {0x13,0x04} | TLS_AES_128_CCM_8_SHA256 | {0x13,0x05} |
|--|--|-------------|-------|------------------------|-------------|------------------------|-------------|------------------------------|-------------|------------------------|-------------|--------------------------|-------------|
| Description  | Value  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_128_GCM_SHA256                                     | {0x13,0x01}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_256_GCM_SHA384                                     | {0x13,0x02}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_CHACHA20_POLY1305_SHA256                               | {0x13,0x03}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_128_CCM_SHA256                                     | {0x13,0x04}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_128_CCM_8_SHA256                                   | {0x13,0x05}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| encrypting a bit stream with a first encryption algorithm; | <p>The standard practices encrypting a bitstream (e.g., bitstream of digital certificate) with a first encryption algorithm (e.g., signature encryption algorithm i.e., SHA256RSA encryption algorithm).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA encryption algorithm) and generates a ciphertext.</p>   |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

As shown below, the accused instrumentality discloses the signature encryption algorithm.

The screenshot shows a Fiddler network traffic capture. On the left, a list of requests is displayed, showing various HTTP and HTTPS connections between the user's browser and external websites like Google and Cloudflare. Request number 23 is highlighted with a red box, indicating it is the selected session. On the right, a detailed view of this selected session is provided. It includes tabs for Transformer, Headers, TextView, SyntaxView, ImageView, HexView, WebView, Auth, Caching, Cookies, Raw, JSON, and XML. The Headers tab shows the secure protocol as "TLS 1.3" and the cipher suite as "TLS\_AES\_256\_GCM\_SHA384". The Transformer tab displays the raw SSL/TLS handshake, including the server certificate information such as the subject (CN=higginbotham.com), version (V3), and issuer (CN=GTS CA 1P5, O=Google Trust Services LLC, C=US). The TextView tab shows the decrypted HTTP request and response messages.

*Source: Fiddler Capture*



The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.  
Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

All handshake messages after the ServerHello are now encrypted. The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

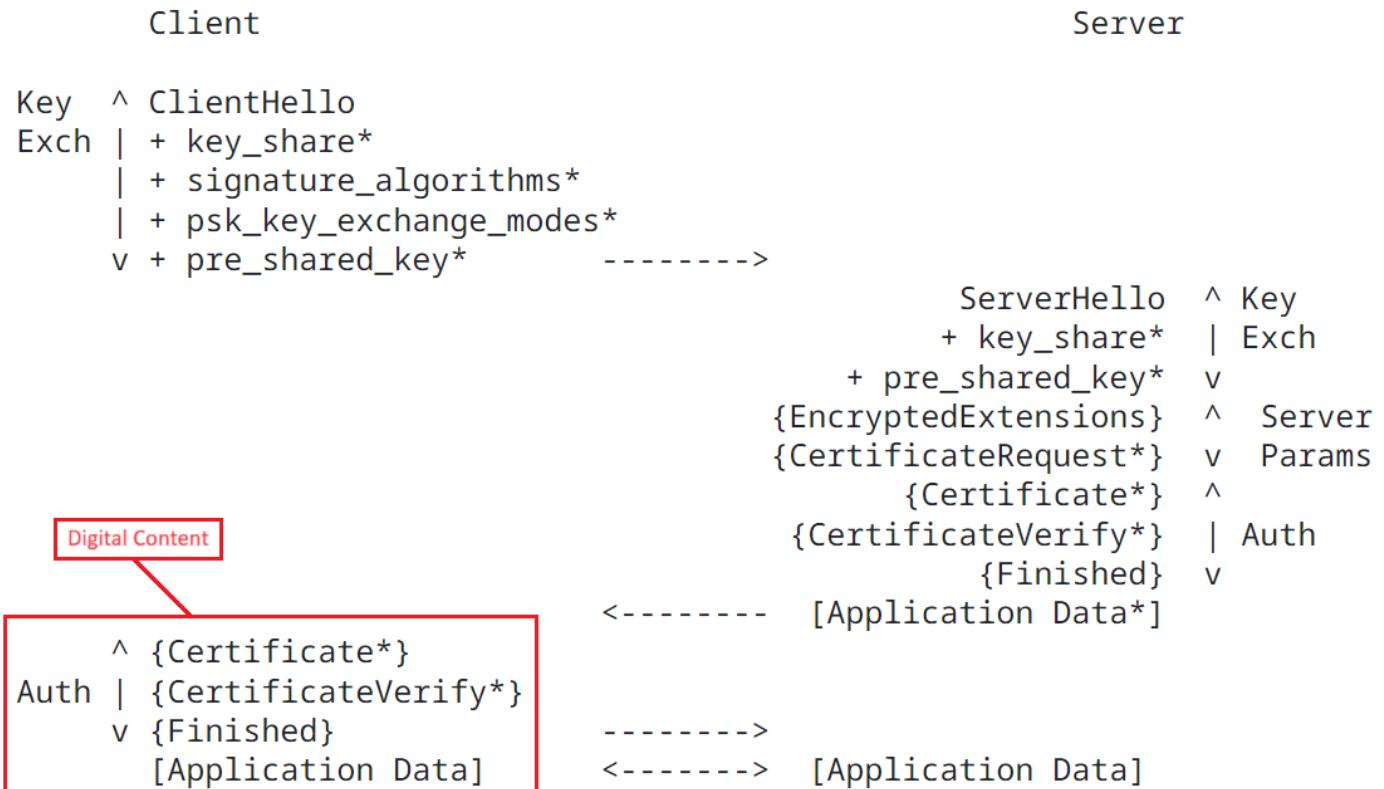
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]](#)\_handshake\_traffic\_secret.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.  
First encryption
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

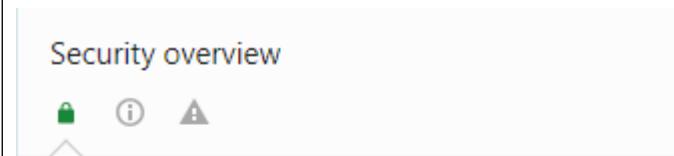
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |   |
|---|---|
|   | <p><b>Introduction</b></p> <p>The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream. Specifically, the secure channel should provide the following properties:</p> <ul style="list-style-type: none"><li>- Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated.<br/>Authentication can happen via asymmetric cryptography (e.g., RSA [RSA], the Elliptic Curve Digital Signature Algorithm (ECDSA) [ECDSA], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032]) or a symmetric pre-shared key (PSK).</li><li>- Confidentiality: Data sent over the channel after establishment is only visible to the endpoints. TLS does not hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.</li><li>- Integrity: Data sent over the channel after establishment cannot be modified by attackers without detection.</li></ul> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446">https://datatracker.ietf.org/doc/html/rfc8446</a></p> |
| associating a first decryption algorithm with the encrypted bit stream; | The standard practices associating a first decryption algorithm (e.g., signature decryption algorithm i.e., SHA256RSA decryption algorithm) with the encrypted bit stream (e.g., encrypted certificate with signature encryption algorithm).<br>The standard practices providing a two-level encryption security for data   |

communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA encryption algorithm) and generates a ciphertext.

The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it. The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate.

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

### The Transport Layer Security (TLS) Protocol Version 1.3

#### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

As shown below, the accused instrumentality discloses the signature decryption algorithm.

[Thumbprint]  
129157F7A5852AD23DF17FE2934E6396DB6B0251

[Signature Algorithm]  
sha256RSA(1.2.840.113549.1.1.11)

[Public Key]  
Algorithm: RSA  
Length: 2048

Key Blob: 30 82 01 0a 02 82 01 01 00 ca b8 66 5f 8a 37 94 14 e1 2f e3 49 cf d7 ad cb 20 8c 54 93 6b af 66 56 02 11 26 db 20 13 00 a0 ce fe ae e9 ac 8d 68 8f a0 56 fd ff b4 b1 4b 9c 54 ff 55 4a 95 76 b9 db 3c 77 5e 9a b1 dd 30 74 36 77 cb 92 12 df 17 62 6d 2a aa 74 3c f6 68 c8 34 8d eb 69 97 9e f2 43 02 fa d7 76 99 c0 bc b3 56 2d 17 ca 39 c8 00 5d 91 c8 1d 9b c9 70 c8 c1 b0 da 40 23 3e 9a fe a9 ed e4 60 ce 15 b0 af 43 a3 07 a5 81 25 dd c3 92 e7 72 9e 27 df bd 01 05 2a b4 60 b8 83 6b fa bc db 97 51 47 a8 2e d0 52 f7 c2 0a 1c 02 96 d2 c7 fd 04 a5 7e 78 04 ee 59 ed 33 9a e4 cc 42 bb 87 36 95 ac 4b 9b 6b 4f 96 4c ed 20 f7 bd 71 4b bc d8 91 81 3c be 56 c8 05 a6 e1 f7 66 85 88 46 91 96 a3 e4 e8 bb 45 f5 b8 78 25 4b 02 cb 23 e6 54 94 ed 94 f6 87 3d 29 55 31 eb 59 e9 a1 54 4c 50 b0 74 2c a7 b6 1e f7 23 02 03 01 00 01  
Parameters: 05 00

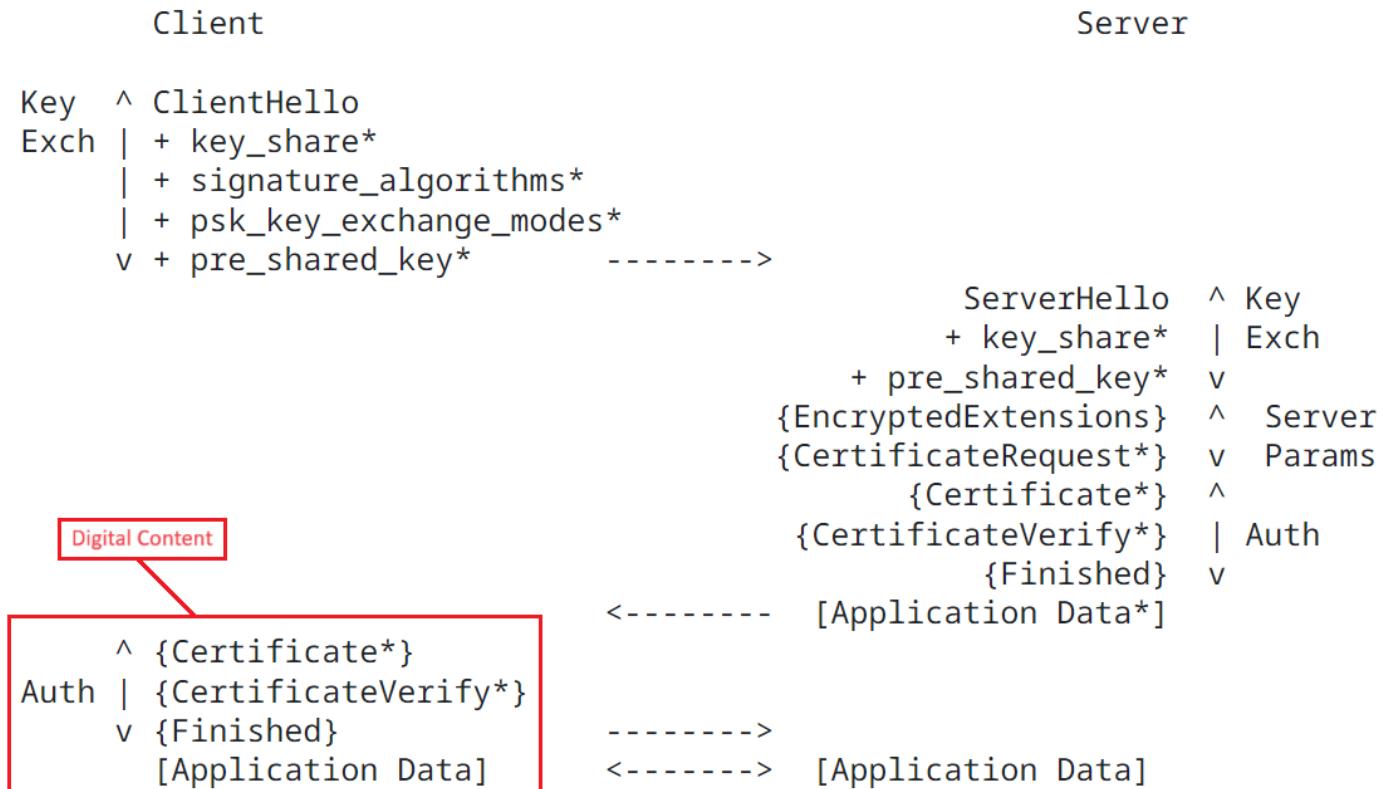
[Extensions]  
\* Key Usage(2.5.29.15):  
Digital Signature, Key Encipherment (a0)

First decryption algorithm

Source: Fiddler Capture

| OID description   |   |
|---|---|
|   | <b>First decryption algorithm identifier</b>  |
| <b>OID:</b>   | {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1)}<br>sha256WithRSAEncryption(11)<br>1.2.840.113549.1.1.11<br>/ISO/Member-Body/US/113549/1/1/11       |
|   | (ASN.1 notation)<br>(dot notation)<br>(OID-IRI notation)  |
| <b>Description:</b>   | Public-Key Cryptography Standards (PKCS) #1 version 1.5 signature algorithm with Secure Hash Algorithm 256 (SHA256) and Rivest, Shamir and Adleman (RSA) encryption |
| <a href="http://oid-info.com/get/1.2.840.113549.1.1.11">http://oid-info.com/get/1.2.840.113549.1.1.11</a>                     |   |
| <pre>-- When the following OIDs are used in an AlgorithmIdentifier, the -- parameters MUST be present and MUST be NULL.</pre> |   |
| <pre>sha224WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 14 }</pre>  |   |
| <pre><u>sha256WithRSAEncryption</u> OBJECT IDENTIFIER ::= { pkcs-1 11 }</pre>   |   |
| <pre>sha384WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 12 }</pre>  |   |
| <pre>sha512WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 13 }</pre>  |   |
| <a href="https://www.ietf.org/rfc/rfc4055.txt">https://www.ietf.org/rfc/rfc4055.txt</a>                                       |   |

Figure 1 below shows the basic full TLS handshake:



#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

All handshake messages after the ServerHello are now encrypted. The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

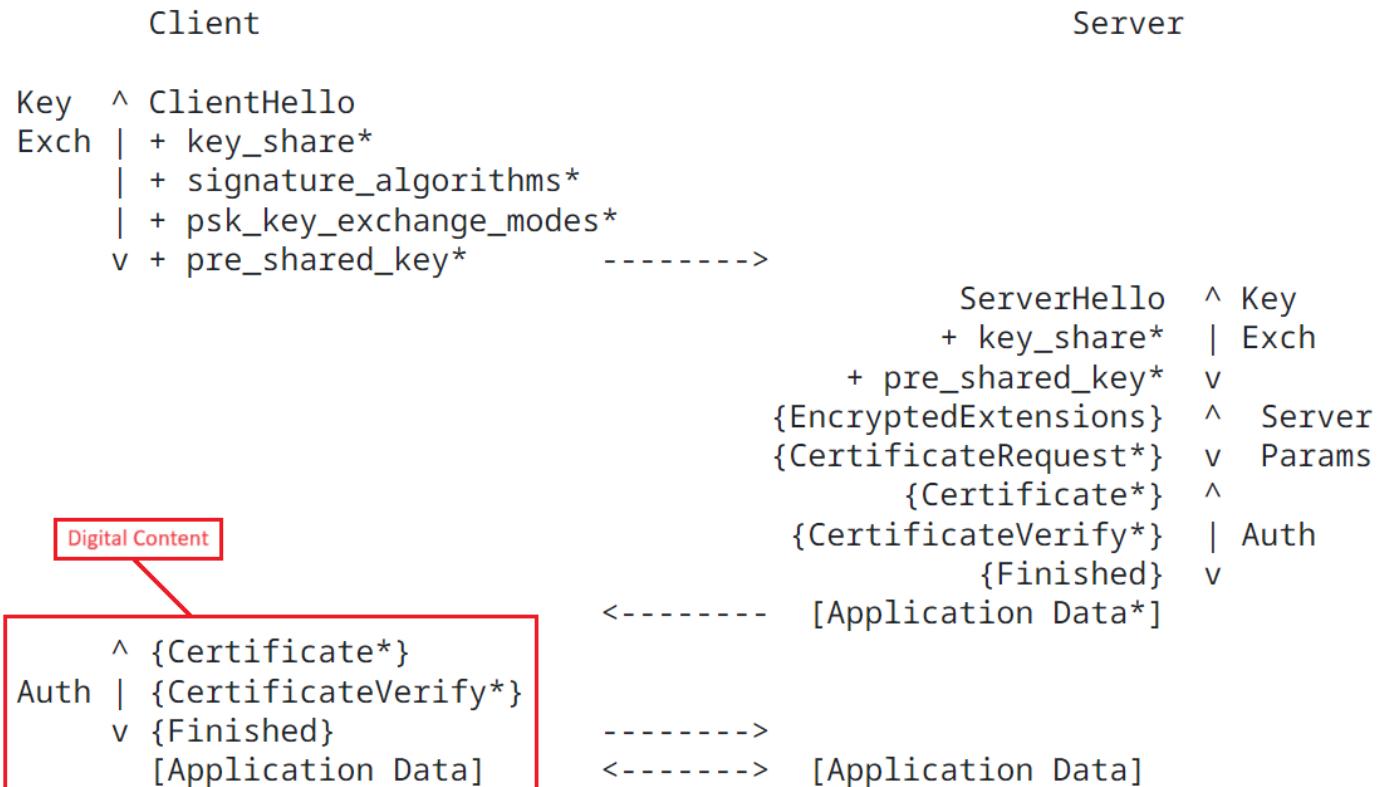
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.  
First encryption
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.3. Certificate Verify

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

First  
decryption  
algorithm  
information

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

As shown below, the receiving party will be able to decrypt the encrypted message with the provided signature decryption algorithm information i.e., SHA-256 RSA decryption algorithm.

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

|  |  |
|--|--|
|  | <p>The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A <i>cryptographic hash function</i> is a function that computes a <i>message authentication code</i> from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that <math>H</math> is a cryptographic hash function. To sign a message <math>m</math>, party A computes <math>h = D_A(H(m))</math> and sends <math>E_B(m, h)</math> to B. Party B now has evidence that A signed <math>m</math> because <math>E_A(h) = H(m)</math>, and A is the only one who could have generated a value <math>h</math> with that property.</p> |
| encrypting both the encrypted bit stream and the first decryption algorithm with a second encryption algorithm to yield a second bit stream; | <p>The standard practices encrypting both the encrypted bit stream (e.g., encrypted digital certificate) and the first decryption algorithm (e.g., signature decryption algorithm) with a second encryption algorithm (e.g., cipher suit selected from one of the AEAD algorithms such as TLS_AES_256_GCM_SHA384, etc.) to yield a second bit stream (e.g., TLS ciphertext bitstream).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA algorithm) and generates a ciphertext.</p> <p>The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it.</p>                           |

The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate. The standard further practices encrypting the authentication message, including the encrypted digital certification and the certificate verify message, with a second decryption algorithm i.e., AEAD algorithm such as TLS\_AES\_256\_GCM\_SHA384, etc.

#### Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.

[View certificate](#)

- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.

- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

| Headers   | TextView   | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm |
|-----------|--|------------|----------|---------|------|---------|-----|------|-----|-----------------------------|
| 00000019  | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 |            |          |         |      |         |     |      |     |                             |
| 00000032  | 77 77 2E 68 69 67 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A |            |          |         |      |         |     |      |     |                             |
| 0000004B  | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55 |            |          |         |      |         |     |      |     |                             |
| 00000064  | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 |            |          |         |      |         |     |      |     |                             |
| 0000007D  | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36 |            |          |         |      |         |     |      |     |                             |
| 00000096  | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48 |            |          |         |      |         |     |      |     |                             |
| 000000AF  | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31 |            |          |         |      |         |     |      |     |                             |
| 000000C8  | 32 36 2E 30 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D |            |          |         |      |         |     |      |     |                             |
| 000000E1  | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E |            |          |         |      |         |     |      |     |                             |
| 000000FA  | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E |            |          |         |      |         |     |      |     |                             |
| 000000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20 |            |          |         |      |         |     |      |     |                             |
| 00000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65 |            |          |         |      |         |     |      |     |                             |
| 00000145  | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72 |            |          |         |      |         |     |      |     |                             |
| 0000015E  | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53 |            |          |         |      |         |     |      |     |                             |
| 00000177  | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69 |            |          |         |      |         |     |      |     |                             |
| 00000190  | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A |            |          |         |      |         |     |      |     |                             |
| 000001A9  | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20 |            |          |         |      |         |     |      |     |                             |
| 000001C2  | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30 |            |          |         |      |         |     |      |     |                             |
| 000001DB  | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37 |            |          |         |      |         |     |      |     |                             |
| 000001F4  | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69 |            |          |         |      |         |     |      |     |                             |
| 0000020D  | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A |            |          |         |      |         |     |      |     |                             |
| 00000226  | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32 |            |          |         |      |         |     |      |     |                             |
| 0000023F  | 20 34 44 20 32 43 20 43 42 20 36 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |            |          |         |      |         |     |      |     |                             |

Source: Fiddler Capture

|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | OC 2A 73 3E B4 85 7A 95   |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     |                           |

Source: Fiddler Capture

The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext

handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [\[RFC5116\]](#), [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

All handshake messages after the ServerHello are now encrypted.  
The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

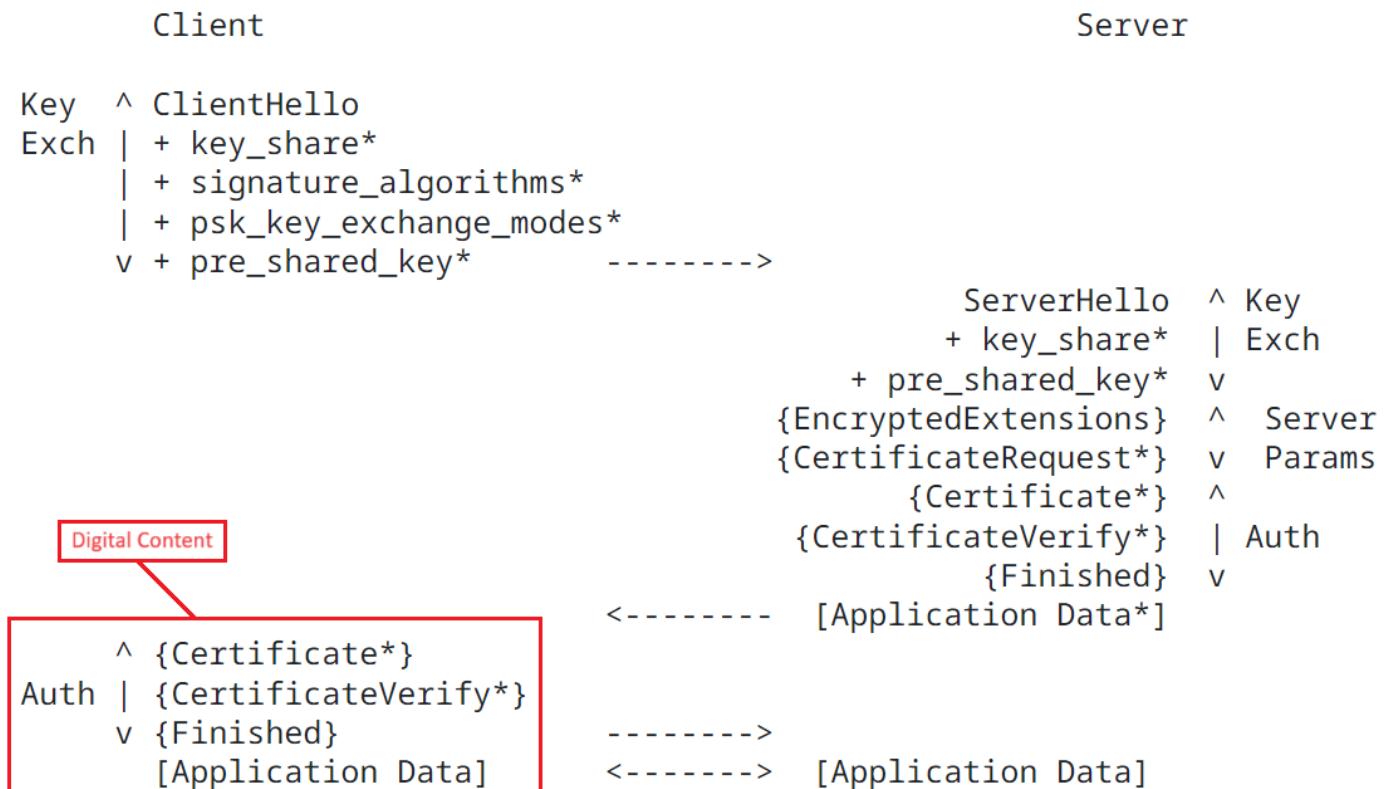
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]](#)\_handshake\_traffic\_secret.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.3. Certificate Verify

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

First  
decryption  
algorithm  
information

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |   |
|---|---|
|   | <p>The <u>"extension_data"</u> field of these extensions contains a <u>SignatureSchemeList</u> value:</p> <pre>enum {     /* RSASSA-PKCS1-v1_5 algorithms */     rsa_pkcs1_sha256(0x0401),     rsa_pkcs1_sha384(0x0501),     rsa_pkcs1_sha512(0x0601),      /* ECDSA algorithms */     ecdsa_secp256r1_sha256(0x0403),     ecdsa_secp384r1_sha384(0x0503),     ecdsa_secp521r1_sha512(0x0603),      /* RSASSA-PSS algorithms with public key OID rsaEncryption */     rsa_pss_rsae_sha256(0x0804),     rsa_pss_rsae_sha384(0x0805),     rsa_pss_rsae_sha512(0x0806),      /* EdDSA algorithms */     ed25519(0x0807),     ed448(0x0808),      /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */     rsa_pss_pss_sha256(0x0809),     rsa_pss_pss_sha384(0x080a),     rsa_pss_pss_sha512(0x080b), }</pre> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| associating a second decryption algorithm with the second bit stream. | <p>The standard practices associating a second decryption algorithm (e.g., cipher suit selected from one of the AEAD algorithms such as TLS_AES_256_GCM_SHA384, etc.) with the second bit stream (e.g., TLS ciphertext bitstream).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA algorithm) and generates a ciphertext.</p>   |

The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it. The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate. The standard further practices encrypting the authentication message, including the encrypted digital certification and the certificate verify message, with a second decryption algorithm i.e., AEAD algorithm such as TLS\_AES\_256\_GCM\_SHA384, etc.

#### Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.

[View certificate](#)

- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.

- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

Transformer Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML

Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.

Secure Protocol: TLS 1.3  
Cipher Suite: TLS\_AES\_256\_GCM\_SHA384

== Server Certificate ======  
[Version]  
V3  
[Subject]  
CN=higginbotham.com  
Simple Name: higginbotham.com  
DNS Name: higginbotham.com  
[Issuer]  
CN=GTS CA 1P5, O=Google Trust Services LLC, C=US  
Simple Name: GTS CA 1P5  
DNS Name: GTS CA 1P5

Source: Fiddler Capture

|          | <p>Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.</p> <p>Secure Protocol: TLS 1.3<br/>Cipher Suite: TLS_AES_256_GCM_SHA384</p> <pre>-- Server Certificate ----- [Version] V3  [Subject] CN=higginbotham.com Simple Name: higginbotham.com DNS Name: higginbotham.com  [Issuer] CN=GTS CA 1P5, O=Google Trust Services LLC, C=US</pre> <p><i>Source: Fiddler Capture</i></p>   |            |          |          |            |          |         |      |         |                           |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
|----------|--|------------|----------|----------|------------|----------|---------|------|---------|---------------------------|------|-----|------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|--------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|
|          | <table border="1"> <thead> <tr> <th>Headers</th><th>TextView</th><th>SyntaxView</th><th>WebForms</th><th>HexView</th><th>Auth</th><th>Cookies</th><th>Raw</th><th>JSON</th><th>XML</th><th>Second bitstream</th></tr> </thead> <tbody> <tr><td>000004C9</td><td>36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>67 98 9F 8B 88 D9 A0 C6 1</td></tr> <tr><td>000004E2</td><td>35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>5 CC 37 BB 51 91 1D 30 CE</td></tr> <tr><td>000004FB</td><td>20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>8D F6 8E 5F 54 C2 5A E3</td></tr> <tr><td>00000514</td><td>38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>9 2C 7B 89 AA BB C5 AF E</td></tr> <tr><td>0000052D</td><td>39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>9 56 10 B8 15 57 94 A1 A1</td></tr> <tr><td>00000546</td><td>20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>0C 2A 73 3E B4 85 7A 95</td></tr> <tr><td>0000055F</td><td>34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>44 34 07 19 C8 33 3F EC 9</td></tr> <tr><td>00000578</td><td>43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>C 45 46 0B 84 C9 1C 22 E1</td></tr> <tr><td>00000591</td><td>20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>3C 9F 87 71 A3 5B C3 72</td></tr> <tr><td>000005AA</td><td>42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>B0 E1 C9 D5 CA 9D C4 69 B</td></tr> <tr><td>000005C3</td><td>30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>0 05 5B 9D E2 A4 2B 56 F5</td></tr> <tr><td>000005DC</td><td>20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>BF 82 84 67 9C C6 65 5F</td></tr> <tr><td>000005F5</td><td>38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>82 3E B2 F0 BB F5 3B AF 0</td></tr> <tr><td>0000060E</td><td>46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>F 86 11 A4 1C 00 5A 46 69</td></tr> <tr><td>00000627</td><td>20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D9 57 AF 31 C6 C2 F4 F0</td></tr> <tr><td>00000640</td><td>33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>38 F7 EC 97 6A 21 78 C4 A</td></tr> <tr><td>00000659</td><td>38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>8 AA B4 85 CE C9 85 67 D7</td></tr> <tr><td>00000672</td><td>20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>7A 0C EB AB 79 1D 83 3A</td></tr> <tr><td>000006BB</td><td>42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>B2 3B 60 D6 A3 C2 01 87 A</td></tr> <tr><td>000006A4</td><td>37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>7 F7 CC B8 A0 E5 1E 1F 55</td></tr> <tr><td>000006BD</td><td>20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>88 34 65 2F BA CA 06 9C</td></tr> <tr><td>000006D6</td><td>33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>3B 17 9D A6 11 B2 39 9F 3</td></tr> <tr><td>000006EF</td><td>34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>4 00 6C CB D0 53 3E 57 16</td></tr> </tbody> </table> <p><i>Source: Fiddler Capture</i></p> <p>The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS</p> |            | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw                       | JSON | XML | Second bitstream | 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |  |  |  |  |  |  |  |  | 67 98 9F 8B 88 D9 A0 C6 1 | 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |  |  |  |  |  |  |  |  | 5 CC 37 BB 51 91 1D 30 CE | 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |  |  |  |  |  |  |  |  | 8D F6 8E 5F 54 C2 5A E3 | 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |  |  |  |  |  |  |  |  | 9 2C 7B 89 AA BB C5 AF E | 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |  |  |  |  |  |  |  |  | 9 56 10 B8 15 57 94 A1 A1 | 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |  |  |  |  |  |  |  |  | 0C 2A 73 3E B4 85 7A 95 | 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |  |  |  |  |  |  |  |  | 44 34 07 19 C8 33 3F EC 9 | 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |  |  |  |  |  |  |  |  | C 45 46 0B 84 C9 1C 22 E1 | 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |  |  |  |  |  |  |  |  | 3C 9F 87 71 A3 5B C3 72 | 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |  |  |  |  |  |  |  |  | B0 E1 C9 D5 CA 9D C4 69 B | 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |  |  |  |  |  |  |  |  | 0 05 5B 9D E2 A4 2B 56 F5 | 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |  |  |  |  |  |  |  |  | BF 82 84 67 9C C6 65 5F | 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |  |  |  |  |  |  |  |  | 82 3E B2 F0 BB F5 3B AF 0 | 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |  |  |  |  |  |  |  |  | F 86 11 A4 1C 00 5A 46 69 | 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |  |  |  |  |  |  |  |  | D9 57 AF 31 C6 C2 F4 F0 | 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |  |  |  |  |  |  |  |  | 38 F7 EC 97 6A 21 78 C4 A | 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |  |  |  |  |  |  |  |  | 8 AA B4 85 CE C9 85 67 D7 | 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |  |  |  |  |  |  |  |  | 7A 0C EB AB 79 1D 83 3A | 000006BB | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |  |  |  |  |  |  |  |  | B2 3B 60 D6 A3 C2 01 87 A | 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |  |  |  |  |  |  |  |  | 7 F7 CC B8 A0 E5 1E 1F 55 | 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |  |  |  |  |  |  |  |  | 88 34 65 2F BA CA 06 9C | 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |  |  |  |  |  |  |  |  | 3B 17 9D A6 11 B2 39 9F 3 | 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |  |  |  |  |  |  |  |  | 4 00 6C CB D0 53 3E 57 16 |
| Headers  | TextView   | SyntaxView | WebForms | HexView  | Auth       | Cookies  | Raw     | JSON | XML     | Second bitstream          |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31   |            |          |          |            |          |         |      |         | 67 98 9F 8B 88 D9 A0 C6 1 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45   |            |          |          |            |          |         |      |         | 5 CC 37 BB 51 91 1D 30 CE |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20   |            |          |          |            |          |         |      |         | 8D F6 8E 5F 54 C2 5A E3   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45   |            |          |          |            |          |         |      |         | 9 2C 7B 89 AA BB C5 AF E  |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31   |            |          |          |            |          |         |      |         | 9 56 10 B8 15 57 94 A1 A1 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20   |            |          |          |            |          |         |      |         | 0C 2A 73 3E B4 85 7A 95   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39   |            |          |          |            |          |         |      |         | 44 34 07 19 C8 33 3F EC 9 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31   |            |          |          |            |          |         |      |         | C 45 46 0B 84 C9 1C 22 E1 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20   |            |          |          |            |          |         |      |         | 3C 9F 87 71 A3 5B C3 72   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42   |            |          |          |            |          |         |      |         | B0 E1 C9 D5 CA 9D C4 69 B |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35   |            |          |          |            |          |         |      |         | 0 05 5B 9D E2 A4 2B 56 F5 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20   |            |          |          |            |          |         |      |         | BF 82 84 67 9C C6 65 5F   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30   |            |          |          |            |          |         |      |         | 82 3E B2 F0 BB F5 3B AF 0 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39   |            |          |          |            |          |         |      |         | F 86 11 A4 1C 00 5A 46 69 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20   |            |          |          |            |          |         |      |         | D9 57 AF 31 C6 C2 F4 F0   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41   |            |          |          |            |          |         |      |         | 38 F7 EC 97 6A 21 78 C4 A |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37   |            |          |          |            |          |         |      |         | 8 AA B4 85 CE C9 85 67 D7 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20   |            |          |          |            |          |         |      |         | 7A 0C EB AB 79 1D 83 3A   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006BB | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41   |            |          |          |            |          |         |      |         | B2 3B 60 D6 A3 C2 01 87 A |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35   |            |          |          |            |          |         |      |         | 7 F7 CC B8 A0 E5 1E 1F 55 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20   |            |          |          |            |          |         |      |         | 88 34 65 2F BA CA 06 9C   |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33   |            |          |          |            |          |         |      |         | 3B 17 9D A6 11 B2 39 9F 3 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36   |            |          |          |            |          |         |      |         | 4 00 6C CB D0 53 3E 57 16 |      |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |

communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

Further, the AEAD encrypted message comprises a ciphertext (e.g., encrypted ciphertext after the encryption by the second encryption algorithm), nonce (e.g., associating second decryption algo), key and associated data. The maximum length of nonce is a cipher suit specific element. The nonce and associated data are utilized in decryption of the AEAD encrypted message.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.2. Authenticated Decryption

Second decryption algorithm

The authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic. A ciphertext C, a nonce N, and associated data A are authentic for key K when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [RFC5116].. [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:  
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

All handshake messages after the ServerHello are now encrypted.  
The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

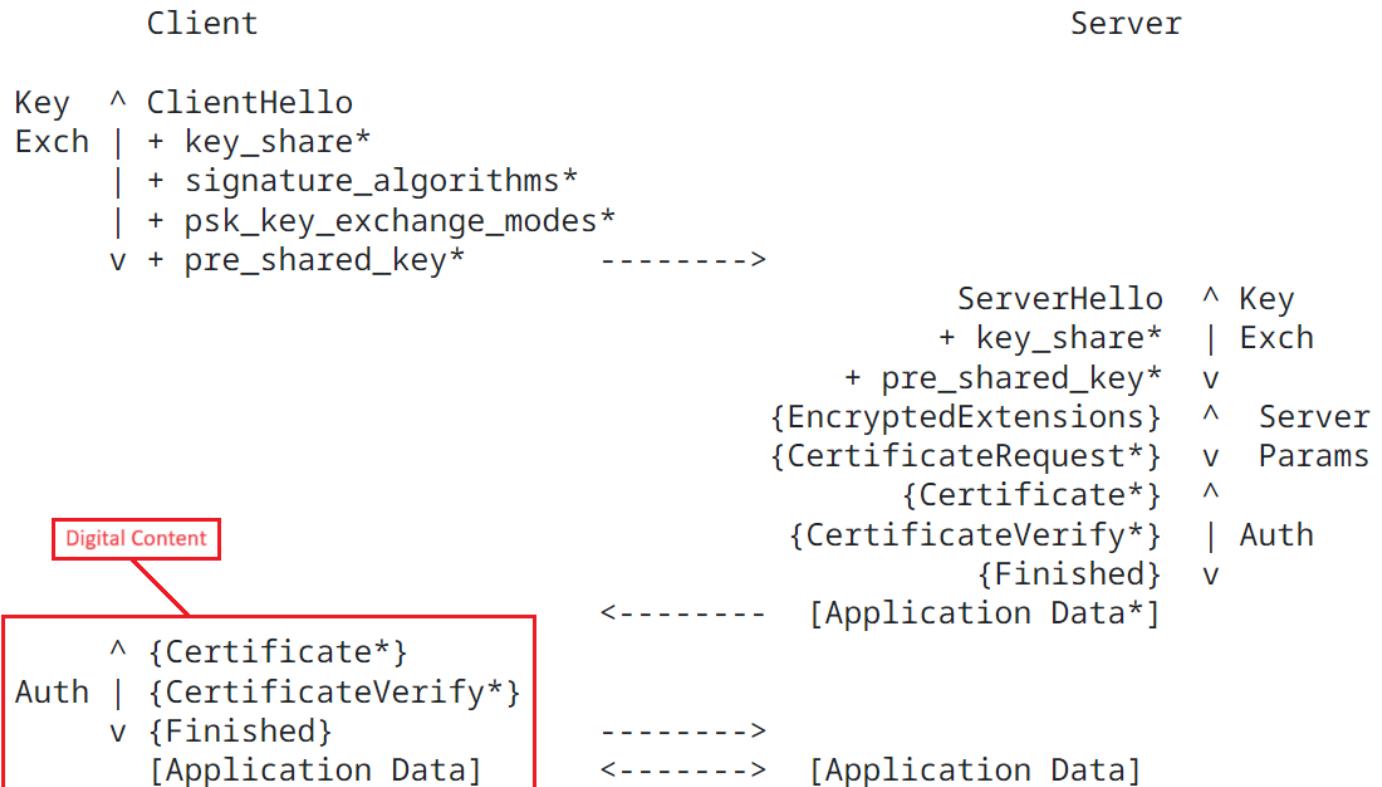
#### **4.4. Authentication Messages**

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block.

These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |  |
|---|--|
|   | <p>The <u>"extension_data"</u> field of these extensions contains a <u>SignatureSchemeList</u> value:</p> <pre>enum {     /* RSASSA-PKCS1-v1_5 algorithms */     rsa_pkcs1_sha256(0x0401),     rsa_pkcs1_sha384(0x0501),     rsa_pkcs1_sha512(0x0601),      /* ECDSA algorithms */     ecdsa_secp256r1_sha256(0x0403),     ecdsa_secp384r1_sha384(0x0503),     ecdsa_secp521r1_sha512(0x0603),      /* RSASSA-PSS algorithms with public key OID rsaEncryption */     rsa_pss_rsae_sha256(0x0804),     rsa_pss_rsae_sha384(0x0805),     rsa_pss_rsae_sha512(0x0806),      /* EdDSA algorithms */     ed25519(0x0807),     ed448(0x0808),      /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */     rsa_pss_pss_sha256(0x0809),     rsa_pss_pss_sha384(0x080a),     rsa_pss_pss_sha512(0x080b), <a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></pre> |
| 20. The system of claim 19, further operable for decrypting the first bit stream and the second | The standard further discloses decrypting the first bit stream (e.g., encrypted digital certificate with signature encryption algorithm i.e., SHA-256 RSA, etc.) and the second bit stream (e.g., a second-level encryption with AEAD encryption algorithm such as TLS AES 256 GCM SHA384, etc.) with the first associated decryption  |

|   |   |
|---|---|
| bit stream with the first associated decryption algorithm and the second associated decryption algorithm wherein the decryption is accomplished by a target unit. | <p>algorithm (e.g., signature decryption algorithm i.e., SHA-256 RSA, etc.) and the second associated decryption algorithm (e.g., cipher suit selected from one of the AEAD decryption algorithms such as TLS_AES_256_GCM_SHA384, etc.) wherein the decryption is accomplished by a target unit (e.g., a server of the accused instrumentality).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA algorithm) and generates a ciphertext.</p> <p>The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it. The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate. The standard further practices encrypting the authentication message, including the encrypted digital certification and the certificate verify message, with a second decryption algorithm i.e., AEAD algorithm such as TLS_AES_256_GCM_SHA384, etc.</p> |
|---|---|

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

The screenshot shows a Fiddler network traffic capture. On the left, a list of requests and responses is displayed, with the 8th request highlighted. This request is a CONNECT tunnel to `www.higginbotham.com:443`. On the right, the SSL certificate details are shown:

| Transformer   | Headers | TextView | SyntaxView | ImageView | HexView | WebView | Auth | Caching | Cookies | Raw | JSON | XML |
|---|---------|----------|------------|-----------|---------|---------|------|---------|---------|-----|------|-----|
| Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list. |         |          |            |           |         |         |      |         |         |     |      |     |
| Secure Protocol: TLS 1.3<br>Cipher Suite: TLS_AES_256_GCM_SHA384  |         |          |            |           |         |         |      |         |         |     |      |     |
| == Server Certificate ======<br>[Version]<br>V3   |         |          |            |           |         |         |      |         |         |     |      |     |
| [Subject]<br>CN=higginbotham.com<br>Simple Name: higginbotham.com<br>DNS Name: higginbotham.com   |         |          |            |           |         |         |      |         |         |     |      |     |
| [Issuer]<br>CN=GTS CA 1P5, O=Google Trust Services LLC, C=US<br>Simple Name: GTS CA 1P5<br>DNS Name: GTS CA 1P5   |         |          |            |           |         |         |      |         |         |     |      |     |

*Source: Fiddler Capture*



[Thumbprint]  
129157F7A5852AD23DF17FE2934E6396DB6B0251

[Signature Algorithm]  
sha256RSA(1.2.840.113549.1.1.11)

First decryption  
algorithm

[Public Key]  
Algorithm: RSA  
Length: 2048

Key Blob: 30 82 01 0a 02 82 01 01 00 ca b8 66 5f 8a 37 94 14 e1 2f e3 49 cf d7 ad cb 20 8c 54 93 6b af 66 56 02 11 26 db 20 13 00 a0 ce fe ae e9 ac 8d 68 8f a0 56 fd ff b4 14 b9 c5 54 ff 55 4a 95 76 b9 db 3c 77 5e 9a b1 dd 30 74 36 77 cb 92 12 df 17 62 6d 2a aa 74 3c f6 68 c8 34 8d eb 69 97 9e f2 43 02 fa d7 76 99 c0 bc b3 56 2d 17 ca 39 c8 00 5d 91 c8 1d 9b c9 70 c8 c1 b0 da 40 23 3e 9a fe a9 ed e4 60 ce 15 b0 af 43 a3 07 a5 81 25 dd c3 92 e7 72 9e 27 df bd 01 05 2a b4 60 b8 83 6b fa bc db 97 51 47 a8 2e d0 52 f7 c2 0a 1c 02 96 d2 c7 fd 04 a5 7e 78 04 ee 59 ed 33 9a e4 cc 42 bb 87 36 95 ac 4b 9b 6b 4f 96 4c ed 20 f7 bd 71 4b bc d8 91 81 3c be 56 c8 05 a6 e1 f7 66 85 88 46 91 96 a3 e4 e8 bb 45 f5 b8 78 25 4b 02 cb 23 e6 54 94 ed 94 f6 87 3d 29 55 31 eb 59 e9 a1 54 4c 50 b0 74 2c a7 b6 1e f7 23 02 03 01 00 01  
Parameters: 05 00

[Extensions]  
\* Key Usage(2.5.29.15):  
Digital Signature, Key Encipherment (a0)

*Source: Fiddler Capture*

| Headers  | Text View  | Syntax View | Web Forms | Hex View | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm   |
|----------|--|-------------|-----------|----------|------|---------|-----|------|-----|---|
| 00000019 | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 |             |           |          |      |         |     |      |     | com:443 HTTP/1.1..Host: www.higginbotham.com:443..Connection: keep-alive..User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36...A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below...Secure |
| 00000032 | 77 77 2E 68 69 67 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A |             |           |          |      |         |     |      |     | Protocol: TLS 1.3.Cipher Suite: TLS_AES_256_GCM_SHA384..Record Layer Version: 3.3 (TLS/1.2).Random:   |
| 0000004B | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55 |             |           |          |      |         |     |      |     | F6 9D 1E 05 3D 58 53 50 C5 38 CB 68 E9 B1 71 BE 0   |
| 00000064 | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 |             |           |          |      |         |     |      |     | 2 77 A7 FA AB 3F CC 1D 97 1B 4C AF AB 7A CD 69."Time": "21-09-1972 08:33:18. SessionID: 5E B3 4B 70 12 4D 2C CB 6A 5B 9A 63 89  |
| 0000007D | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36 |             |           |          |      |         |     |      |     |   |
| 00000096 | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48 |             |           |          |      |         |     |      |     |   |
| 000000AF | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31 |             |           |          |      |         |     |      |     |   |
| 000000CB | 32 36 2E 30 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D |             |           |          |      |         |     |      |     |   |
| 000000E1 | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E |             |           |          |      |         |     |      |     |   |
| 000000FA | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E |             |           |          |      |         |     |      |     |   |
| 00000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20 |             |           |          |      |         |     |      |     |   |
| 0000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65 |             |           |          |      |         |     |      |     |   |
| 00000145 | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72 |             |           |          |      |         |     |      |     |   |
| 0000015E | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53 |             |           |          |      |         |     |      |     |   |
| 00000177 | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69 |             |           |          |      |         |     |      |     |   |
| 00000190 | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A |             |           |          |      |         |     |      |     |   |
| 000001A9 | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20 |             |           |          |      |         |     |      |     |   |
| 000001C2 | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30 |             |           |          |      |         |     |      |     |   |
| 000001DB | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37 |             |           |          |      |         |     |      |     |   |
| 000001F4 | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69 |             |           |          |      |         |     |      |     |   |
| 0000020D | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A |             |           |          |      |         |     |      |     |   |
| 00000226 | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32 |             |           |          |      |         |     |      |     |   |
| 0000023F | 20 34 44 20 32 43 20 43 42 20 36 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |             |           |          |      |         |     |      |     |   |

*Source: Fiddler Capture*

|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 39 2C 7B 89 AA BB C5 AF E |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | 0C 2A 73 3E B4 85 7A 95   |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |

*Source: Fiddler Capture*

Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.

Secure Protocol: TLS 1.3  
 Cipher Suite: TLS\_AES\_256\_GCM\_SHA384

-- Server Certificate -----  
 [Version]  
 V3

[Subject]  
 CN=higginbotham.com  
 Simple Name: higginbotham.com  
 DNS Name: higginbotham.com

[Issuer]  
 CN=GTS CA 1P5, O=Google Trust Services LLC, C=US

*Source: Fiddler Capture*

The standard defines four record message types, including a handshake message type.

The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

Further, the AEAD encrypted message comprises a ciphertext (e.g., encrypted ciphertext after the encryption by the second encryption algorithm), nonce (e.g., associating second decryption algo), key and associated data. The maximum length of nonce is a cipher suit specific element. The nonce and associated data are utilized in decryption of the AEAD encrypted message.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.2. Authenticated Decryption

Second decryption algorithm

The authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic. A ciphertext C, a nonce N, and associated data A are authentic for key K when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [RFC5116].. [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:  
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

All handshake messages after the ServerHello are now encrypted.  
The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

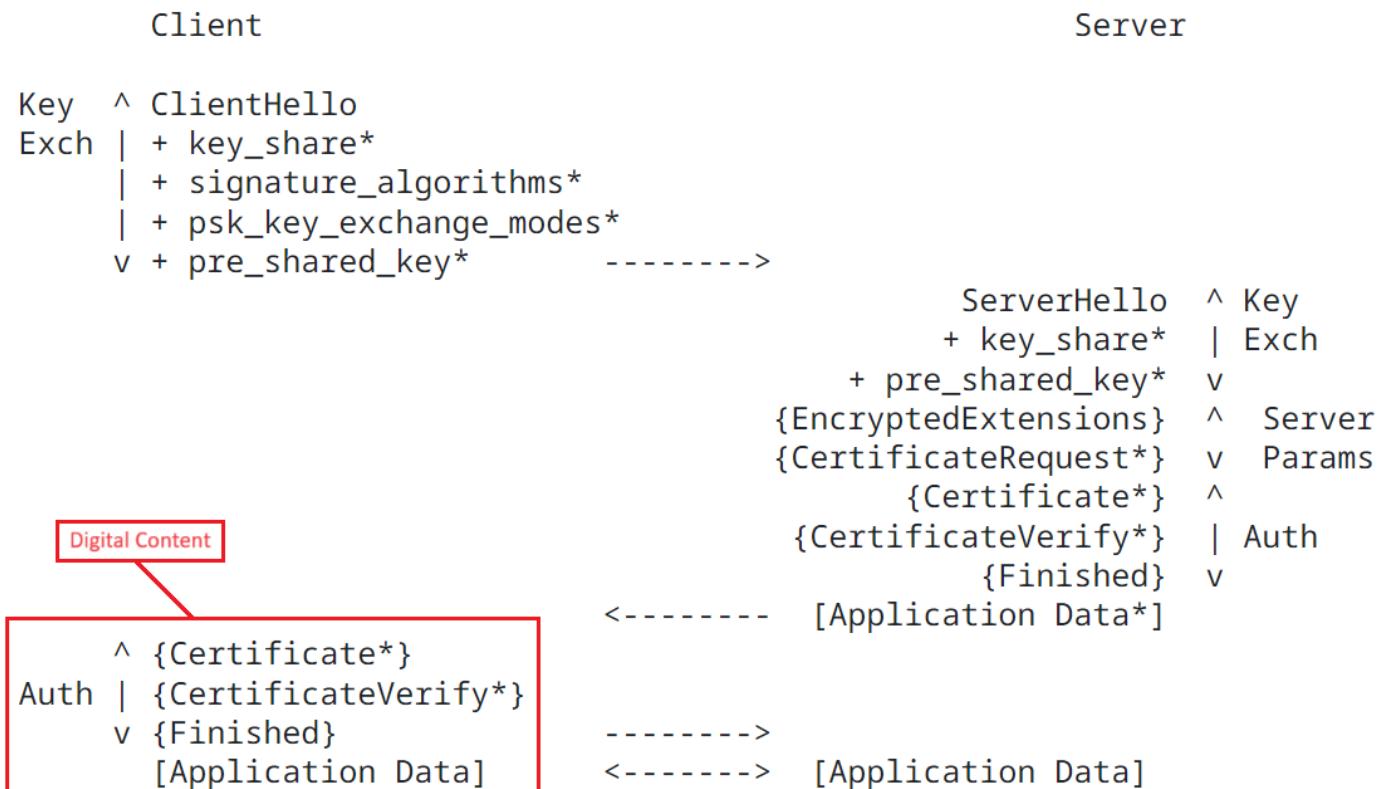
#### **4.4. Authentication Messages**

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block.

These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

As shown below, the receiving party will be able to decrypt the encrypted message with the provided signature decryption algorithm information i.e., SHA-256 RSA decryption algorithm.

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

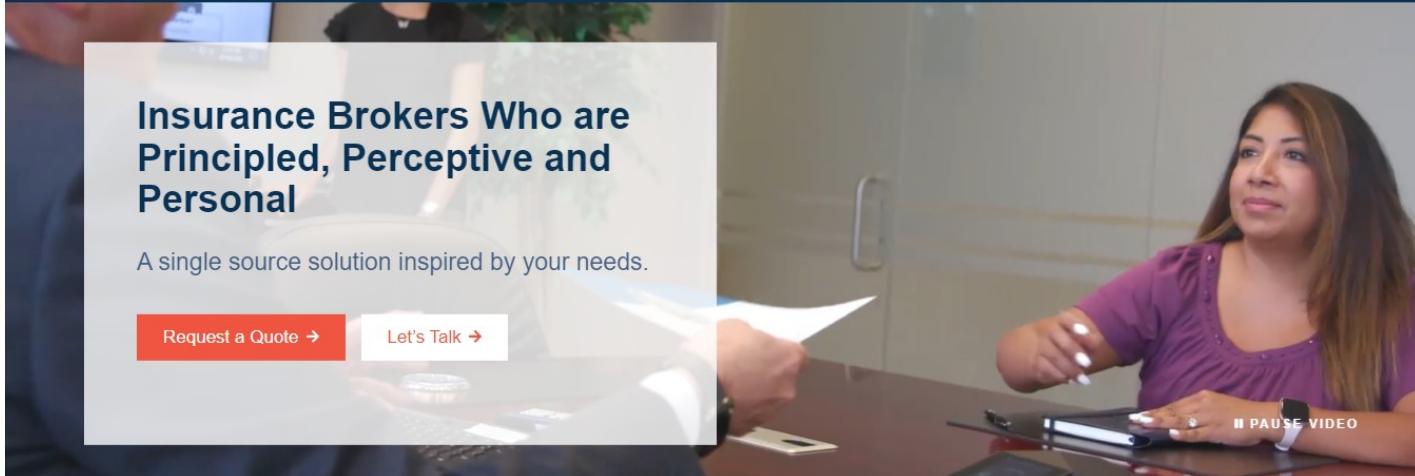
First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

|   |  |
|---|--|
|   | <p>The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A <i>cryptographic hash function</i> is a function that computes a <i>message authentication code</i> from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that <math>H</math> is a cryptographic hash function. To sign a message <math>m</math>, party A computes <math>h = D_A(H(m))</math> and sends <math>E_B(m, h)</math> to B. Party B now has evidence that A signed <math>m</math> because <math>E_A(h) = H(m)</math>, and A is the only one who could have generated a value <math>h</math> with that property.</p> |
| 21. The system of claim 20, wherein the decrypting is done using a key associated with each decryption algorithm. | The standard practices the method such that the decrypting is done using a key (e.g., decryption key) associated with each decryption algorithm (e.g., signature decryption algorithm such as SHA-256RSA, etc., and AEAD decryption algorithm such as TLS_AES_256_GCM_SHA384, etc.).   |

The Higginbotham Difference About Us Careers Blog Locations Become a Higginbotham Partner  Report a Claim

 Higginbotham\* Business Insurance Employee Benefits HR Services Financial Services Personal Insurance Captive Insurance Let's Talk



Insurance Brokers Who are Principled, Perceptive and Personal

A single source solution inspired by your needs.

Request a Quote → Let's Talk →

<https://www.higginbotham.com/>

II PAUSE VIDEO

The screenshot shows the Google Play Store search results for the query "higginbotham". The top result is the "Higginbotham FSA" app, which is described as "Higginbotham Wex Health Mobile". It features a red box around its listing. Below the app are three promotional screenshots of the mobile application interface, showing account details, account activity, and expense management.

<https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US>

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP-Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 18B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB 89 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
dreasel(0xdada) 00
```

*Source: Fiddler Capture*

As shown below, the signature decryption algorithm utilizes a private key for a first decryption and the AEAD decryption algorithm uses a key K. Both the decryption techniques are decrypting using their respective associated keys.

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

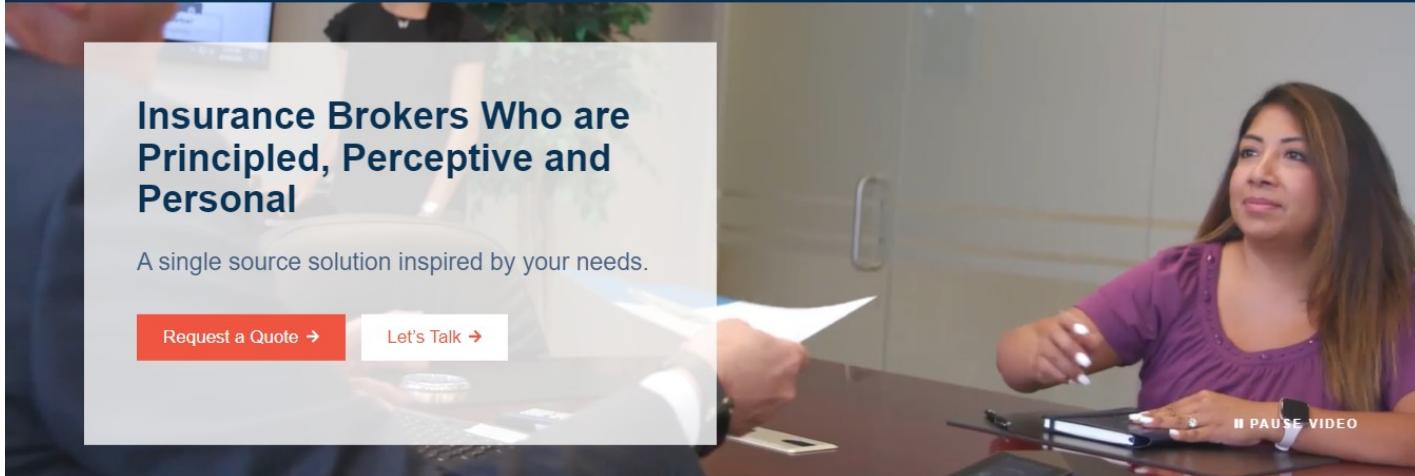
| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |  |
|---|--|
|   | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 22. The system of claim 21, wherein the key is resident in hardware of the target unit or the key is retrieved from a server. | The standard utilized by the accused instrumentality practices the method such that the key is resident in hardware (e.g., stored in a memory storage of the server such as a database, RAM, etc.) of the target unit (e.g., server of the accused instrumentality) or the key is retrieved from a server.   |

The Higginbotham Difference About Us Careers Blog Locations Become a Higginbotham Partner  Report a Claim

 Higginbotham\* Business Insurance Employee Benefits HR Services Financial Services Personal Insurance Captive Insurance Let's Talk



Insurance Brokers Who are Principled, Perceptive and Personal

A single source solution inspired by your needs.

Request a Quote → Let's Talk →

<https://www.higginbotham.com/>

II PAUSE VIDEO

The screenshot shows the Google Play Store interface. At the top, there is a search bar with the text "higginbotham". Below the search bar, there are three categories: "Apps & games" (selected), "Movies & TV", and "Books". A red box highlights the first search result, which is the "Higginbotham FSA" app. The app's icon is a blue circle with a white letter "H". It has a download count of "1K+" and is rated "Everyone". A large "Install" button is visible. To the right of the search results, there are three promotional cards for the app:

- "View your account(s) and link to resources from 'I Want To'" showing a smartphone screen with account details.
- "Check your account activity anytime, anywhere" showing a smartphone screen with account activity history.
- "Manage expenses from a consolidated dashboard" showing a smartphone screen with expense management features.

At the bottom of the page, there is a purple bar with the URL <https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US> and a "Privacy - Terms" link. Below the URL, there is a block of hex code and the text "Source: Fiddler Capture".

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP-Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 58 6E
2D 8A 88 BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 FB 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
driase (0xdada) 00
```

Source: Fiddler Capture



## **Server hardware guide: Architecture, products and management**

### **3. Random access memory**



RAM is the main type of memory in a computing system.

RAM holds the software instructions and data needed by the processor, along with any output from the processor, such as data to be moved to a storage device. Thus, RAM works very closely with the processor and must match the processor's incredible speed and performance. This kind of fast memory is usually termed dynamic RAM, and several DRAM variations are available for servers.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>



## **Server hardware guide: Architecture, products and management**

### **4. Hard disk drive**



This hardware is responsible for reading, writing and positioning of the hard disk, which is one technology for data storage on server hardware. Developed at IBM in 1953, the [hard disk drive \(HDD\)](#) has evolved over time from the size of a refrigerator to the standard 2.5-inch and 3.5-inch form factors.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>

As shown below, the server comprises a memory storage to store messages for establishing secure TLS communication. the standard discloses multiple signature encryption algorithms for a first encryption and multiple AEAD encryption algorithms for the second encryption. A signature decryption algorithm utilizes a private key for decrypting the first bitstream encrypted with the signature encryption and an AEAD decryption algorithm uses a key K for decrypting the second bitstream encrypted with the AEAD encryption. Both the decryption techniques are decrypting using their respective associated keys. A server must have a storage to store information pertaining to these algorithms and their corresponding keys such as private key, Key K, etc., to establish secure TLS communication with a client.

Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. This is necessary for the ClientHello storage mechanism described in Section 8.2 because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.

<https://datatracker.ietf.org/doc/html/rfc8446#>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

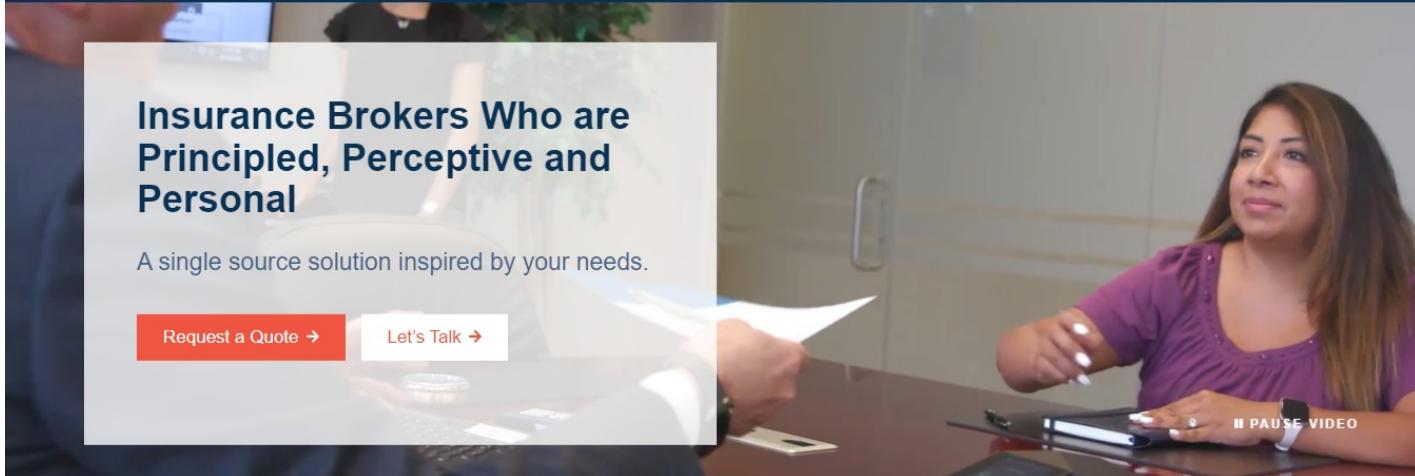
| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |  |
|---|--|
|   | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 23. The system of claim 22, wherein the key is contained in a key data structure. | The standard utilized by the accused instrumentality practices the method such that the key (e.g., private key, Key K, etc.) is contained in a key data structure (e.g., data structure).  |

The Higginbotham Difference About Us Careers Blog Locations Become a Higginbotham Partner  Report a Claim

 **Higginbotham** \* Business Insurance Employee Benefits HR Services Financial Services Personal Insurance Captive Insurance Let's Talk



Insurance Brokers Who are Principled, Perceptive and Personal

A single source solution inspired by your needs.

[Request a Quote →](#) [Let's Talk →](#)

<https://www.higginbotham.com/>

II PAUSE VIDEO

The screenshot shows the Google Play Store interface. A search bar at the top contains the query "higginbotham". Below the search bar, there are three category tabs: "Apps & games" (selected), "Movies & TV", and "Books". To the right of the search bar are a magnifying glass icon, a close button, and a user profile icon.

The main search results are displayed below. The first result is a card for the "Higginbotham FSA" app, which is part of the "Higginbotham Wex Health Mobile" suite. The card includes the app's logo, a download count of "1K+", a rating of "Everyone", and a large "Install" button. A red rectangular box highlights this card.

Below the app card, there are three promotional screenshots of the app's mobile interface. The first screenshot shows the "My Accounts" screen with sections for FSA, HSA, and DependentCare. The second screenshot shows the "Account Activity" screen with a table of transactions. The third screenshot shows the "Dashboard" screen with a summary of expenses and a "CREATE NEW EXPENSE" button.

At the bottom of the page, there is a purple horizontal bar containing a URL: <https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US>. To the right of the URL is a small blue icon and the text "Privacy - Terms".

A code snippet is also visible on the page:

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP-Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 88 BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB 89 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
dreasel(0xada) 00
```

*Source: Fiddler Capture*

The accused instrumentality utilizes a server to establish a secure TLS communication with a client. The server must comprise a memory storage and store data according to a data structure to implement the standard efficiently.



## **Server hardware guide: Architecture, products and management**

### **3. Random access memory**



RAM is the main type of memory in a computing system.

RAM holds the software instructions and data needed by the processor, along with any output from the processor, such as data to be moved to a storage device. Thus, RAM works very closely with the processor and must match the processor's incredible speed and performance. This kind of fast memory is usually termed dynamic RAM, and several DRAM variations are available for servers.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>



## Server hardware guide: Architecture, products and management



### 4. Hard disk drive

This hardware is responsible for reading, writing and positioning of the hard disk, which is one technology for data storage on server hardware. Developed at IBM in 1953, the [hard disk drive \(HDD\)](#) has evolved over time from the size of a refrigerator to the standard 2.5-inch and 3.5-inch form factors.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>

A data structure is a specialized format for organizing, processing, retrieving and storing data.

There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose. Data structures make it easy for users to access and work with the data they need in appropriate ways. Most importantly, data structures frame the organization of information so that machines and humans can better understand it.

In computer science and computer programming, a data structure may be selected or designed to store data for the purpose of using it with various algorithms. In some cases, the algorithm's basic operations are tightly coupled to the data structure's design. Each data structure contains information about the data values, relationships between the data and -- in some cases -- functions that can be applied to the data.

<https://www.techtarget.com/searchdatamanagement/definition/data-structure>

As shown below, the server comprises a memory storage to store messages for establishing secure TLS communication. the standard discloses multiple signature encryption algorithms for a first encryption and multiple AEAD encryption algorithms for the second encryption. A signature decryption algorithm utilizes a private key for decrypting the first bitstream encrypted with the signature encryption and an AEAD decryption algorithm uses a key K for decrypting the second bitstream encrypted with the AEAD encryption. Both the decryption techniques are decrypting using their respective associated keys. A server must have a storage to store information pertaining to these algorithms and their corresponding keys such as private key, Key K, etc., to establish secure TLS communication with a client.

Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. This is necessary for the ClientHello storage mechanism described in [Section 8.2](#) because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.

<https://datatracker.ietf.org/doc/html/rfc8446#>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|  |  |
|--|--|
|  | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 29. The system of claim 21, wherein each encryption algorithm is a symmetric key system or an asymmetric key system. | The standard practices the method such that each encryption algorithm (e.g., signature encryption algorithm i.e., SHA256RSA, etc., and AEAD encryption algorithm i.e., TLS_AES_256_GCM_SHA384, etc.) is a symmetric key system (e.g., AEAD encryption algorithm, etc.) or an asymmetric key system (e.g., signature encryption algorithm). <p>As shown below, the server comprises a memory storage to store messages for</p>  |

establishing secure TLS communication. the standard discloses multiple signature encryption algorithms for a first encryption and multiple AEAD encryption algorithms for the second encryption. A signature decryption algorithm utilizes a private key for decrypting the first bitstream encrypted with the signature encryption and an AEAD decryption algorithm uses a key K for decrypting the second bitstream encrypted with the AEAD encryption. The standard defines the signature encryption algorithm as an asymmetric cryptography algorithm and the AEAD encryption algorithm as the symmetric cryptography algorithm.

Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. This is necessary for the ClientHello storage mechanism described in Section 8.2 because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.

<https://datatracker.ietf.org/doc/html/rfc8446#>

Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated. Authentication can happen via asymmetric cryptography (e.g., RSA [RSA], the Elliptic Curve Digital Signature Algorithm (ECDSA) [ECDSA], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032]) or a symmetric pre-shared key (PSK).

<https://datatracker.ietf.org/doc/html/rfc8446#section-4>

cipher\_suites: A list of the symmetric cipher options supported by the client, specifically the record protection algorithm (including secret key length) and a hash to be used with HKDF, in descending order of client preference. Values are defined in [Appendix B.4](#). If the list contains cipher suites that the server does not recognize, support, or wish to use, the server MUST ignore those cipher suites and process the remaining ones as usual. If the client is attempting a PSK key establishment, it SHOULD advertise at least one cipher suite indicating a Hash associated with the PSK.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in Section 3.2, and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

|   |  |
|---|--|
|   | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 30. The system of claim 21, further operable for associating a first Message Authentication Code (MAC) or first digital signature with each | The standard practices associating a first Message Authentication Code (MAC) (e.g., message authentication code with hashing function) or first digital signature with each encrypted bit stream (e.g., encrypted bit stream with the signature encryption algorithm i.e., SHA256RSA, etc., and encrypted bitstream with the AEAD encryption algorithm i.e., TLS_AES_256_GCM_SHA384, etc.). <p>As shown below, the standard discloses a hashing function with each of the encryption</p>   |

encrypted bit stream. algorithm. It performs a message authentication code with the utilized hashing function.

```

SignedCertTimestamp (RFC6962) empty
ALPN          h2, http/1.1
signature_algs ecdsa secp256r1 sha256, rsa_pss_rsae sha256, rsa_pkcs1 sha256, ecdsa secp384r1 sha384, rsa_pss_rsae sha384, rsa_pkcs1_sha384, rsa_pss_rsae_sha512
0x001b 02 00 02
supported_groups grease [0x5a5a], unknown [0x6399], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]
key_share      04 ED 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 32 5D 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BC C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85
7A 95 44 34 07 19 C8 33 3F EC 9C 45 46 0B 84 C9 1C 22 E1 3C 9F 87 71 A3 5B C3 72 80 E1 C9 5D CA 9D C4 69 B0 05 5B 9D E2 4B 2B 56 F5 BF 82 84 67 9C C6 65 5F 82 3E B2 F0 BB F5 3B AF 0F 86 11 A4 1C 00 5A 46 68 D9 57 AF 31 C6 C2 F4 F0 38 F1
EC 97 6A 21 78 C4 A8 AA B4 85 CE 93 85 67 D7 7A 0C EB AB 79 1D 83 3A B2 3B 60 D6 A3 C2 01 87 A7 F7 CC B8 A0 E5 1E 1F 35 88 34 65 2F BA CA 06 9C 3B 17 9D A6 11 B2 39 9F 34 00 6C DB D0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E 2F
C0 61 1D A0 54 53 73 A1 99 23 8C 39 BA A0 C6 73 7A 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A B5 2C 21 80 CA 1F E2 AC B4 73 A5 FD EB 7B 6D B5 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F E0 75 BA 39 AA 0B 66 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F
23 C1 C3 B1 5E E8 03 13 61 7C 42 DC 1C 3F F0 AB 47 A3 00 28 E6 A7 C1 C1 26 7B D7 FB 9F E9 09 14 5A E3 B6 5A 5B 9B D2 D5 47 0D 6B 7B 4A 71 6D 7E 01 37 F8 E0 A3 1E 58 6B D6 D2 C0 26 53 B8 9B B0 09 DF 5A 20 22 E9 A3 DD 00 9E 98 F0 4C A0
11 99 AD C0 48 DA D1 37 3B 2A AF 72 43 91 BD 66 6E 23 56 C7 5A E0 5E 74 33 2F 3A 86 40 07 F3 D7 95 B1 38 70 23 05 30 1B 5E A5 C7 07 13 84 9A 90 B3 66 B5 11 A9 A6 72 3C 15 DC EC 94 3E 55 93 83 64 C0 BF 79 2F C3 22 6C B1 C4 AB 5B 12
83 A0 B0 8F 26 98 2F 47 47 7B 2B 10 49 21 80 5F 14 53 0B 4E DA 90 E3 A0 82 98 C8 89 6E 63 TA 6A AA A7 E6 C8 08 2D 4B 10 BA 70 A0 3A 93 D1 75 07 59 1C 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 99 3C E8 A3 A7 F0 73 D4 FA 81 9F 99 BF 70 B7 A1
08 BB 58 A8 C2 2D C1 FB BB B7 23 89 CB F0 92 B8 A6 B3 E4 6C 6B 59 20 88 B5 53 12 EB 49 EB A3 13 B3 78 A5 F1 3A 70 86 8B 8B 96 96 7F 23 9C 37 19 0C C0 9C 0B 14 8F 67 78 4A F5 03 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 A4 AB
CA 6B 3C A4 85 1D 6E 70 74 88 C7 9D B9 8B 4D DD 4B 39 6D AB 00 9C 52 17 C3 C4 AD F6 10 A0 DE 0A B5 86 C7 49 SD 94 3A C5 39 95 02 BC 1E A2 73 6A 2D 16 1C 54 31 68 1D A6 C9 85 C3 A0 12 3B 1C 72 10 60 3E 3B 23 6A 68 92 DF CC 86 91 E7 9E
2B 72 01 FC C4 34 07 61 05 82 53 94 E6 2C CC 30 74 A3 A7 08 4D 91 77 TA 14 5C 4D 4F E2 68 66 94 AD 20 34 22 4F 29 2B 29 37 42 25 08 53 R2 F4 CD A9 19 3A 44 8A 23 98 EC 22 E9 01 24 6D 29 8B DC 08 B0 45 B9 45 88 43 BA E4 99 3A 32 23 32 6C
66 31 26 FC 06 97 C2 2E 7D 39 B9 AA 17 69 3C 94 66 FA C3 3B C9 E4 5C 75 BC 59 8B 60 14 A7 26 83 FF 72 2C 0D F7 4F E9 88 B0 33 67 6D F3 C2 C0 1A 5B 69 2F 61 48 53 94 9A F6 67 00 03 7A 8C 8E BC 4E 4C 2C 49 E0 82 8C DA 88 AF DA 6A 6D B5 08
9D CA A9 A5 5A F2 91 95 3C 70 5A 3C B0 B2 AD CE 61 A9 8F C4 54 15 8B 3C A9 31 21 D8 28 CD AC 99 09 A2 37 B4 46 53 2F AF E4 AF E8 F2 55 CC FC 4F 4B 78 A4 FC 28 64 10 17 C6 61 71 2F 90 25 26 E0 C5 38 2A C6 4C A4 99 9E F8 D6 3B 1C
48 C0 AE DC B0 B6 1C 85 B3 75 4F B4 5A 14 3F C0 90 CD E6 B7 18 12 6E 34 52 C1 B5 A3 1B E4 17 72 1F F8 25 20 88 21 6B B1 A3 C3 E4 A4 2E 53 6C CC 0A 82 32 07 76 C5 23 8C C0 40 07 00 C0 B0 C2 4A 48 D3 B7 13 00 11 C5 88 73 26 F2 EA 96 15 55
CC 3E EB 0F 80 28 93 D4 49 B4 1D 52 18 44 B8 86 BD F2 BC 1E 43 34 23 C8 57 4B 65 2F E6 29 97 C8 39 39 BD E7 0F 2D C2 AF 53 DA AA 56 C1 7D 4B 93 54 D9 67 C5 D9 F5 1E 7B 86 49 57 45 B5 58 56 84 32 42 B5 39 60 62 40 C2 96
B1 24 9A DA B4 8D FD 8A 9C SC C0 87 1E 19 12 FD FA 32 DB 69 C3 B3 7B 76 84 15 70 C9 69 73 7A F6 A6 D7 A1 21 4C F0 63 06 42 37 TD F3 3A 85 05 EF D6 02 FD F1 6D 4A 3C 55 AB BA 8A 0F 45 A6 5C C8 53 19 F6 A1 75 06 90 A0 26 2D A3 90
9B AB 18 53 2C E9 0F 05 83 3D 0E 49 76 33 D8 4D 8E 72 29 2F FB 51 3B 42 5F 76 C4 AE 54 45 E6 91 8D 2B 45 F0 9A 1E 65 C0 3A D7 99 F5 AE F1 9D E9 45 D3 A3 C3 DB 7A 74 E0 3D ED 7F 6D 00 01 20 FA D2 73 2A 7D BB 08 13 72 9B 38 A0 89 F4
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP - Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 0F 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 62 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 88 3C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 FB 90 B6 F6 1A 4E 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 C6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
osrae (0x3a3a) 00

```

*Source: Fiddler Capture*

SignedCertTimestamp (RFC6962) empty  
ALPN h2, http/1.1  
signature algs ecdsa secp256r1 sha256, rsa\_pss\_rsae\_sha256, rsa\_pkcs1\_sha256, ecdsa secp384r1 sha384, rsa\_pss\_rsae\_sha384, rsa\_pkcs1\_sha384, rsa\_pss\_rsae\_sha512, rsa\_pkcs1\_sha512  
0x001b 02 00 02  
supported\_groups grease [0x5a5a], unknown [0x6399], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]  
key\_share 04 ED 5A 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 32 5D 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 E8 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85  
7A 95 44 34 07 19 C8 33 3F BC 93 45 08 B4 94 C1 22 E1 93 49 8F 77 13 A5 B3 72 B0 C1 C9 D5 CA 69 80 05 9B 92 E4 2A 28 56 F9 6C 65 5F 82 E2 B0 F5 B3 AB 0F 86 11 A4 1C 00 54 46 89 D9 57 AF 31 C6 C2 F4 F0 38 F1  
EC 97 6A 21 28 C4 A8 AA B4 85 C9 C8 95 67 D7 7A 0C EB AB 79 1D 83 3A B2 3B 60 D6 A3 C2 01 87 A7 F7 CC 88 A0 E5 1E 0F 55 88 34 65 2F BA CA 06 9C 38 17 9D A6 11 B2 39 9F 34 00 6C CB D0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E 2F  
C0 61 1D A0 54 55 73 A1 99 23 8C 39 BA A0 C6 73 7A 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A B5 2C 21 80 CA 1F E2 AC 84 73 A5 FD EB 7B 6D B3 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F E0 75 BA 39 AA 0B 86 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F5  
23 C1 03 0B 5E 08 13 61 C2 4D 1C 3F B0 47 A3 00 28 E6 A7 C1 C1 26 7B D7 FB 9F E9 09 14 5A E3 B6 5A 5B 9B D2 D5 47 0D 6B 7B 4A 71 6D 7E 01 37 F0 E0 A3 1E 58 6B 06 D2 C0 26 53 B3 98 B0 09 DF 5A 20 22 E9 A3 DD 00 E6 98 F0 4C A0  
11 99 AD C0 48 DA D1 37 3B 2A AF T2 43 91 BD 66 8E 23 56 C7 5A E0 5E 74 33 BF 3A 86 40 07 F3 7D 95 B1 38 70 23 05 30 1E 5E A5 C7 07 13 84 84 A9 80 B3 66 B5 11 A9 A8 61 72 3C 15 DC EC 94 3E 55 93 83 64 CO BF 79 2F C3 22 6C B1 C4 AB 58 12  
83 A0 B0 8F 26 98 2F 47 7B 20 81 5F 14 53 04 BE A0 E3 82 98 C8 89 6E 73 AA 7A A8 1E 7C 08 2D 4B 10 BA 70 A0 3A 93 01 75 07 91 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 98 C8 E3 A3 A7 F0 73 D4 FA 81 9F 99 BF 77 B7 A1  
08 BB 58 A8 CC 2D C1 FB BR B7 23 89 CF 90 B2 AB 65 B3 4F 6C 6B 59 20 88 B5 12 49 FB 4A 13 B3 7A 85 F1 3A 70 86 8B 96 96 7F 23 3C 79 19 0C C0 3C 08 14 4F 67 78 4A F5 03 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 A4 AB  
CA 6B 3C 4A 85 1D 6E 70 74 86 C7 9D B9 8B 4D DD 4B 39 6D AB 00 9C 52 17 C3 C3 AD F6 10 A0 DE 0A B5 86 C7 49 5D 94 3A C5 39 95 02 BC 1E A2 73 6A 2D 16 1C 54 31 68 1D A6 C9 B5 C3 A0 12 3B 1C 72 10 60 3E 2B 3D 6A 68 92 DF CC 86 91 E7 98  
2B 72 01 TB C4 34 07 61 59 34 93 H4 36 2C CC 30 74 A3 7A 08 47 91 77 14 5C 28 6B 66 94 20 34 22 4F 29 2B 29 37 23 08 53 92 F4 CD A5 19 3A 44 8A 23 98 EC 22 E9 01 24 6D 29 BB 8C 45 88 43 8A E4 99 3A 32 23 6C 66  
31 26 FC 06 97 C2 2E TD 39 B9 AA 17 69 3C 94 66 FA C3 CB 9E 45 7C 5B 59 8B 60 14 A7 26 83 FF 72 2C 0D F4 E9 88 03 33 67 FD F3 C2 C0 1A 5B 69 2F 61 48 53 94 FA C6 67 00 03 7A 8C BE 4C 4C 2C 49 E0 82 8C DA 88 AF DA 6D B5 08  
9D CA 09 5A M2 91 50 7A 5C 3C D2 0D CE 61 B4 54 15 8B 3C A9 32 21 D8 2C AC 99 09 A2 37 B4 46 53 2F AE 4F E8 F2 55 CC FC 4F 4B 78 A4 FC 28 64 10 17 C6 61 71 2F 90 25 26 E0 C5 38 2A C6 4C A4 99 9E F8 D8 3B 1C  
48 CO AE DC B0 86 1C 85 83 75 4F BA 54 14 3F C0 90 CD E6 87 18 12 6E 34 52 C1 B5 A3 1B E4 17 72 1F F8 25 20 88 21 6B 1A 3C E4 4A 2E 53 6C CO DA 82 32 07 W6 C5 23 8C CO 40 07 00 CO B0 C2 4A 48 D3 B7 13 00 11 C5 88 73 26 F2 EA 96 15 55  
C0 3E EB OF 08 28 93 D4 49 B4 1D 52 18 44 88 88 6D F5 BC AD 88 60 BF F2 BC 1E 43 34 23 C8 57 48 65 2F E6 29 97 C8 39 3B ED 70 F2 D2 C2 AF 53 DA AA 56 C1 7D 48 93 54 D9 67 C5 D9 F5 1E 7B 86 49 57 45 85 58 56 84 32 42 85 39 60 62 40 C2 96  
B1 24 9A DA B4 8D FD 8A 5C C0 87 19 12 FD FA 32 FD 69 C3 B3 7B 76 84 15 70 C9 69 73 7A F6 A6 D7 A1 21 4C F0 63 06 42 37 FD F3 A3 B5 D0 85 EF D6 02 FD F1 6D 4A 3C 55 AB BA 8A 0F 45 A6 5C C8 53 19 F6 A1 75 06 90 A0 26 2D A3 90  
9B AB 18 53 2C E9 OF 05 83 3D DE 49 76 33 D8 4D 8E 72 29 2F FB 51 3B 42 5F 76 C4 AE 54 45 E6 91 8D 2B 45 F0 9A 1E 65 C0 3A D7 99 F5 AE F1 9D E9 45 D3 A3 CD DB 7A 74 03 D7 ED F7 6D 00 01 20 FA D2 73 2A 7D BB 08 13 72 9B 38 A0 89 F4  
7A 38 23 01 54 D1 F6 FB F9 92 32 D7 C6 C4 34 24  
  
extended\_master\_secret empty  
ec\_point\_formats uncompressed [0x0]  
status\_request OCSP - Implicit Responder  
psk\_key\_exchange\_modes 01 01  
renegotiation\_info 00  
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15  
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 C3 58 6E  
2D 8A 8B BD F9 05 D4 63 E1 FC 35 82 4F 90 98 5C AB 90 32 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 F8 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C  
E4 C8 A4 D5 F3 C8 D8 F9 SA DF C3 0A 9F 27 E4 C6 05 53 F5 83 38 B9 5A  
  
server\_name www.higginbotham.com  
grease [0x3a3a] 00

*Source: Fiddler Capture*

| Headers  | TextView  | SyntaxView   | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm |
|----------|---|--|----------|---------|------|---------|-----|------|-----|-----------------------------|
| 00000019 | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77    | com:443 HTTP/1.1..Host: www.higginbotham.com:443..Connection: keep-alive..User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36... |          |         |      |         |     |      |     |                             |
| 00000032 | 77 77 2E 68 69 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A       | .A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below...Secure   |          |         |      |         |     |      |     |                             |
| 0000004B | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55    | Protocol: TLS 1.3.Cipher Suite: TLS_AES_256_GCM_SHA384..Record Layer Version: 3.3 (TLS/1.2).Random:  |          |         |      |         |     |      |     |                             |
| 00000064 | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57    | F6 9D 1E 05 3D 58 53 50  |          |         |      |         |     |      |     |                             |
| 0000007D | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36    | C5 38 CB 68 E9 B1 71 BE 0  |          |         |      |         |     |      |     |                             |
| 00000096 | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48    | 2 77 A7 FA AB 3F CC 1D 97  |          |         |      |         |     |      |     |                             |
| 000000AF | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31    | 1B 4C AF AB 7A CD 69."Time": "21-09-1972 08:33:18.   |          |         |      |         |     |      |     |                             |
| 000000C8 | 32 36 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D          | SessionID: ME B3 4B 70 12  |          |         |      |         |     |      |     |                             |
| 000000E1 | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E    | 4D 2C CB 6A 5B 9A 63 89  |          |         |      |         |     |      |     |                             |
| 000000FA | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E    |  |          |         |      |         |     |      |     |                             |
| 00000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20    |  |          |         |      |         |     |      |     |                             |
| 0000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65    |  |          |         |      |         |     |      |     |                             |
| 00000145 | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72    |  |          |         |      |         |     |      |     |                             |
| 0000015E | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53    |  |          |         |      |         |     |      |     |                             |
| 00000177 | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69    |  |          |         |      |         |     |      |     |                             |
| 00000190 | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A    |  |          |         |      |         |     |      |     |                             |
| 000001A9 | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20    |  |          |         |      |         |     |      |     |                             |
| 000001C2 | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30    |  |          |         |      |         |     |      |     |                             |
| 000001DB | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37    |  |          |         |      |         |     |      |     |                             |
| 000001F4 | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69    |  |          |         |      |         |     |      |     |                             |
| 0000020D | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A    |  |          |         |      |         |     |      |     |                             |
| 00000226 | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32    |  |          |         |      |         |     |      |     |                             |
| 0000023F | 20 34 44 20 32 43 20 43 42 20 36 35 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |  |          |         |      |         |     |      |     |                             |

|          | <i>Source: Fiddler Capture</i>   |          |            |          |         |      |         |     |      |     |                           |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 9 2C 7B 89 AA BB C5 AF E  |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | 0C 2A 73 3E B4 85 7A 95   |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |

*Source: Fiddler Capture*

|   |   |
|---|---|
|   | <p>The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A <i>cryptographic hash function</i> is a function that computes a <i>message authentication code</i> from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that <math>H</math> is a cryptographic hash function. To sign a message <math>m</math>, party <math>A</math> computes <math>h = D_A(H(m))</math> and sends <math>E_B(m, h)</math> to <math>B</math>. Party <math>B</math> now has evidence that <math>A</math> signed <math>m</math> because <math>E_A(h) = H(m)</math>, and <math>A</math> is the only one who could have generated a value <math>h</math> with that property.</p> <p><a href="https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf">https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf</a></p> <p>The list of supported symmetric encryption algorithms has been pruned of all algorithms that are considered legacy. Those that remain are all Authenticated Encryption with Associated Data (AEAD) algorithms. The cipher suite concept has been changed to separate the authentication and key exchange mechanisms from the record protection algorithm (including secret key length) and a <u>hash to be used with both the key derivation function and handshake message authentication code (MAC)</u>.</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-4">https://datatracker.ietf.org/doc/html/rfc8446#section-4</a></p> |
| 37. A computer storage device for a recursive | The accused instrumentality utilizes a computer storage device (e.g., a memory of the server of the accused instrumentality) for a recursive security protocol (e.g., TLS 1.3   |

security protocol for protecting digital content, comprising instructions executable by a processor for performing the steps of:

security protocol) for protecting digital content (e.g., digital certificate related to the accused instrumentality), comprising instructions executable by a processor (e.g., a processor of the server of the accused instrumentality).

The accused instrumentality utilizes TLS 1.3 security protocol (hereinafter “the standard”) for communicating content such as digital certificate, application data, etc., with a client. The standard provides a two-level encryption security. It encrypts a plaintext with a first encryption technique and generates a ciphertext. Further, it encrypts the ciphertext with a second encryption technique i.e., recursive encryption security.



The screenshot shows the Google Play Store interface. At the top, there is a search bar with the text "higginbotham". Below the search bar, there are three categories: "Apps & games" (selected), "Movies & TV", and "Books". On the right side of the header, there is a user profile icon and a question mark icon.

The main content area displays search results. The first result is highlighted with a red box and labeled "Higginbotham FSA Higginbotham Wex Health Mobile". It includes a brief description: "Save time and hassles with the Higginbotham FSA mobile app", a circular icon with a stylized letter "H", a download count of "1K+", a rating of "Everyone", and an "Install" button.

Below the main result, there are three promotional cards:

- "View your account(s) and link to resources from 'I Want To'" (showing a smartphone screen with account details like FSA, HSA, and DependentCare).
- "Check your account activity anytime, anywhere" (showing a smartphone screen with account activity details).
- "Manage expenses from a consolidated dashboard" (showing a smartphone screen with expense management details).

At the bottom of the page, there is a purple horizontal bar containing the URL <https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US>. To the right of the URL, there is a small blue square icon with a white "P" and the text "Privacy - Terms".

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

As shown below, the accused instrumentality utilizes a two-level algorithm security. It utilizes the SHA256RSA encryption algorithm as a first encryption algorithm i.e., signature encryption algorithm and the TLS\_AES\_256\_GCM\_SHA384 encryption algorithm as a second encryption algorithm i.e., AEAD encryption algorithm.

A screenshot of the Fiddler network traffic capture interface. On the left, a list of requests and responses is shown, with the 23rd item highlighted. This item shows a request to 'www.higginbotham.com' via HTTPS. On the right, the details pane displays the secure session information. It shows the 'Secure Protocol' as 'TLS 1.3' and the 'Cipher Suite' as 'TLS\_AES\_256\_GCM\_SHA384'. Below this, the 'Server Certificate' details are listed, including the CN (Common Name) as 'higginbotham.com', the Simple Name as 'higginbotham.com', and the DNS Name as 'higginbotham.com'. The issuer information is also provided, stating 'CN=GTS CA 1P5, O=Google Trust Services LLC, C=US', 'Simple Name: GTS CA 1P5', and 'DNS Name: GTS CA 1P5'. The interface includes tabs for Transformer, Headers, TextView, SyntaxView, ImageView, HexView, WebView, Auth, Caching, Cookies, Raw, JSON, and XML.

*Source: Fiddler Capture*

|  |   |
|--|---|
|  | <pre> SignedCertTimestamp (RFC5962) empty ALPN h2, http/1.1 signature_algs ecdsa_secp256r1_sha256,rsa_pss_rsae_sha256,rsa_pkcs1_sha256,ecdsa_secp384r1_sha384,rsa_pss_rsae_sha384,rsa_pkcs1_sha384,rsa_pss_rsae_sha512,rsa_pkcs1_sha512 0x001b 02 00 02 supported_groups grease [0x5a5a], unknown [0x6399], x25519 [0xd1], secp256r1 [0x17], secp384r1 [0x18] key_share 04 ED 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 39 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85 7A 95 44 34 07 19 C8 33 3F EC 9C 45 46 08 84 C9 1C 22 E1 3C 9F 87 71 A3 5B C3 72 80 E1 C9 D5 CA 9D C4 69 B0 05 5B 9D E2 A4 2B 56 F5 BF 82 84 67 9C C6 65 5F 82 3E B2 F0 B8 F5 3B AF 0F 86 11 A4 1C 00 5A 46 68 D9 57 AF 31 C6 C2 F4 F0 38 F1 EC 97 6A 21 78 C4 A8 AA B4 85 CE C9 85 67 D7 7A 0C EB AB 79 1D 83 3A B2 3B 60 D6 A3 C2 01 87 A7 F7 CC B8 A0 E5 1E 1F 55 88 34 65 2B BA CA 06 9C 3B 17 9D A6 11 B2 39 9F 34 00 6C CB D0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E 2F C0 61 1D A0 54 55 73 A1 99 23 8C 39 BA A0 C6 73 A8 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A 85 2C 21 80 CA 1F E2 AC B4 73 A5 FD EB 7B 6D 85 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F E0 75 BA 39 AA 0B 66 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F1 11 99 AD 04 48 DA 37 3B 7A 2F 47 7B 2B 10 49 20 5F 14 53 0B 4E DA 90 E3 A0 82 98 C8 89 6E 63 TA 8A AA A7 E6 08 2D 4B 10 8A 70 A0 3A 93 D1 75 07 59 1C 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 99 3C E8 A3 A7 F0 73 D4 FA 81 9F 99 BF 70 B7 A1 23 C1 C3 B1 E8 05 13 61 C7 42 DC 1C 3F F0 AB 47 A3 00 28 E6 A7 C1 C1 26 7B D7 FB 9F E9 09 14 5A E3 B4 5A 5B 9B D2 D5 47 0D 6B 7B 4A 71 6D 7E 01 37 F8 E0 A3 1E 58 6B D6 D2 C0 26 53 8B 9B 00 09 DF 5A 20 22 E9 A3 DD 00 9E 98 F0 4C A0 83 A0 B0 8F 26 98 2F 47 47 7B 2B 10 49 20 5F 14 53 0B 4E DA 90 E3 A0 82 98 C8 89 6E 63 TA 8A AA A7 E6 08 2D 4B 10 8A 70 A0 3A 93 D1 75 07 59 1C 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 99 3C E8 A3 A7 F0 73 D4 FA 81 9F 99 BF 70 B7 A1 extended_master_secret empty ec_point_format uncompressed [0x0] status_request OCSP- Implicit Responder psk_key_exchange_modes 01 01 renegotiation_info 00 0xfe0d 00 00 01 00 01 BB 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15 F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 AB BC B3 30 C6 42 13 6B B6 62 52 TA 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E 2D 8A 8B BD F9 05 06 D4 63 E1 FC 35 82 4F 90 98 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 FB 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 48 E0 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 89 5A server_name www.higginbotham.com orase (0x3a3a) 00 </pre> <p><b>Digital certificate</b></p> <p><b>First encryption algorithm</b></p> <p><b>Source: Fiddler Capture</b></p> |
|--|---|

[Thumbprint]  
129157F7A5852AD23DF17FE2934E6396DB6B0251

[Signature Algorithm]  
sha256RSA(1.2.840.113549.1.1.11)

First decryption  
algorithm

[Public Key]  
Algorithm: RSA  
Length: 2048

Key Blob: 30 82 01 0a 02 82 01 01 00 ca b8 66 5f 8a 37 94 14 e1 2f e3 49 cf d7 ad cb 20 8c 54 93 6b af 66 56 02 11 26 db 20 13 00 a0 ce fe ae e9 ac 8d 68 8f a0 56 fd ff b4 14b 9c 54 f5 55 4a 95 76 b9 db 3c 77 5e 9a b1 dd 30 74 36 77 cb 92 12 df 17 62 6d 2a aa 74 3c f6 68 c8 34 8d eb 69 97 9e f2 43 02 fa d7 76 99 c0 bc b3 56 2d 17 ca 39 c8 00 5d 91 c8 1d 9b c9 70 c8 c1 b0 da 40 23 3e 9a fe a9 ed e4 60 ce 15 b0 af 43 a3 07 a5 81 25 dd c3 92 e7 72 9e 27 df bd 01 05 2a b4 60 b8 83 6b fa bc db 97 51 47 a8 2e d0 52 7f c2 0a 1c 02 96 d2 c7 fd 04 a5 7e 78 04 ee 59 ed 33 9a e4 cc 42 bb 87 36 95 ac 4b 9b 6b 4f 96 4c ed 20 f7 bd 71 4b bc d8 91 81 3c be 56 c8 05 a6 e1 7f 66 85 88 46 91 96 a3 e4 e8 bb 45 f5 b8 78 25 4b 02 cb 23 e6 54 94 ed 94 f6 87 3d 29 55 31 eb 59 e9 a1 54 4c 50 b0 74 2c a7 b6 1e 7f 23 02 03 01 00 01  
Parameters: 05 00

[Extensions]  
\* Key Usage(2.5.29.15):  
Digital Signature, Key Encipherment (a0)

*Source: Fiddler Capture*

| Headers  | Text View  | Syntax View | Web Forms | Hex View | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm   |
|----------|--|-------------|-----------|----------|------|---------|-----|------|-----|---|
| 00000019 | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 |             |           |          |      |         |     |      |     | com:443 HTTP/1.1..Host: www.higginbotham.com:443..Connection: keep-alive..User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36...A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below...Secure |
| 00000032 | 77 77 2E 68 69 67 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A |             |           |          |      |         |     |      |     | Protocol: TLS 1.3.Cipher Suite: TLS_AES_256_GCM_SHA384..Record Layer Version: 3.3 (TLS/1.2).Random:   |
| 0000004B | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55 |             |           |          |      |         |     |      |     | F6 9D 1E 05 3D 58 53 50 C5 38 CB 68 E9 B1 71 BE 0   |
| 00000064 | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 |             |           |          |      |         |     |      |     | 2 77 A7 FA AB 3F CC 1D 97 1B 4C AF AB 7A CD 69."Time": "21-09-1972 08:33:18. SessionID: 5E B3 4B 70 12 4D 2C CB 6A 5B 9A 63 89  |
| 0000007D | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36 |             |           |          |      |         |     |      |     |   |
| 00000096 | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48 |             |           |          |      |         |     |      |     |   |
| 000000AF | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31 |             |           |          |      |         |     |      |     |   |
| 000000CB | 32 36 2E 30 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D |             |           |          |      |         |     |      |     |   |
| 000000E1 | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E |             |           |          |      |         |     |      |     |   |
| 000000FA | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E |             |           |          |      |         |     |      |     |   |
| 00000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20 |             |           |          |      |         |     |      |     |   |
| 0000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65 |             |           |          |      |         |     |      |     |   |
| 00000145 | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72 |             |           |          |      |         |     |      |     |   |
| 0000015E | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53 |             |           |          |      |         |     |      |     |   |
| 00000177 | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69 |             |           |          |      |         |     |      |     |   |
| 00000190 | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A |             |           |          |      |         |     |      |     |   |
| 000001A9 | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20 |             |           |          |      |         |     |      |     |   |
| 000001C2 | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30 |             |           |          |      |         |     |      |     |   |
| 000001DB | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37 |             |           |          |      |         |     |      |     |   |
| 000001F4 | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69 |             |           |          |      |         |     |      |     |   |
| 0000020D | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A |             |           |          |      |         |     |      |     |   |
| 00000226 | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32 |             |           |          |      |         |     |      |     |   |
| 0000023F | 20 34 44 20 32 43 20 43 42 20 36 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |             |           |          |      |         |     |      |     |   |

*Source: Fiddler Capture*

|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 39 2C 7B 89 AA BB C5 AF E |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | 0C 2A 73 3E B4 85 7A 95   |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |

*Source: Fiddler Capture*

Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.

Secure Protocol: TLS 1.3  
 Cipher Suite: TLS\_AES\_256\_GCM\_SHA384

-- Server Certificate -----  
 [Version]  
 V3

[Subject]  
 CN=higginbotham.com  
 Simple Name: higginbotham.com  
 DNS Name: higginbotham.com

[Issuer]  
 CN=GTS CA 1P5, O=Google Trust Services LLC, C=US

*Source: Fiddler Capture*

As shown below, the server of the accused instrumentality comprises a processor to

execute instructions and a memory storage to store instructions for performing the operations defined by the standard.

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP - Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 88 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 B6 FB 61 AE 44 A1 D9 62 78 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
orase (0x3a3a) 00
```

*Source: Fiddler Capture*



## Server hardware guide: Architecture, products and management



### 2. Processor

The CPU -- or simply processor -- is a complex micro-circuitry device that serves as the foundation of all computer operations. It supports hundreds of possible commands hardwired into hundreds of millions of transistors to process low-level software instructions -- microcode -- and data and derive a desired logical or mathematical result. The processor works closely with memory, which both holds the software instructions and data to be processed as well as the results or output of those processor operations.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>



## **Server hardware guide: Architecture, products and management**

### **3. Random access memory**



RAM is the main type of memory in a computing system.

RAM holds the software instructions and data needed by the processor, along with any output from the processor, such as data to be moved to a storage device. Thus, RAM works very closely with the processor and must match the processor's incredible speed and performance. This kind of fast memory is usually termed dynamic RAM, and several DRAM variations are available for servers.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>



## Server hardware guide: Architecture, products and management

### 4. Hard disk drive



This hardware is responsible for reading, writing and positioning of the hard disk, which is one technology for data storage on server hardware. Developed at IBM in 1953, the [hard disk drive \(HDD\)](#) has evolved over time from the size of a refrigerator to the standard 2.5-inch and 3.5-inch form factors.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>

Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. This is necessary for the ClientHello storage mechanism described in [Section 8.2](#) because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.

<https://datatracker.ietf.org/doc/html/rfc8446#>

The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

All handshake messages after the ServerHello are now encrypted. The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

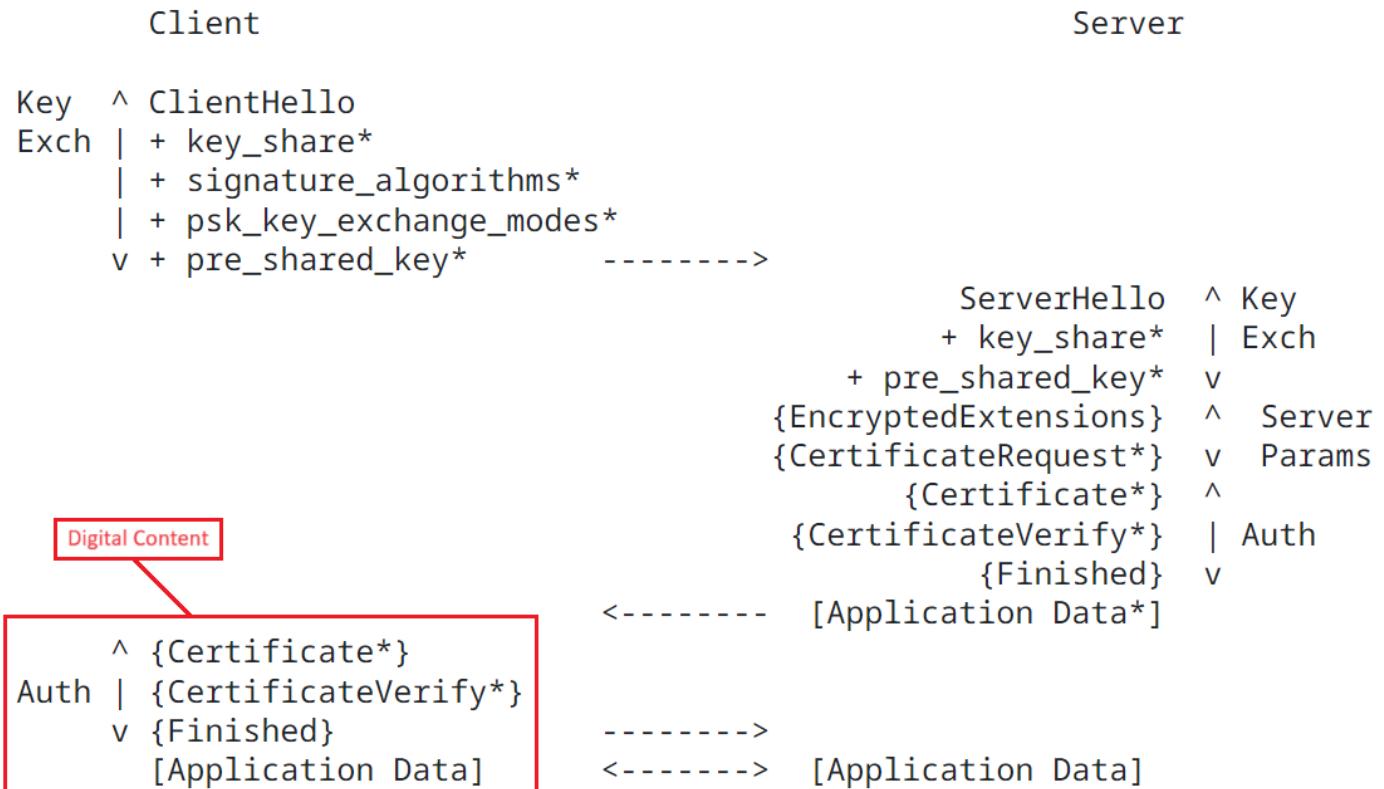
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]](#)\_handshake\_traffic\_secret.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## Introduction

The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream. Specifically, the secure channel should provide the following properties:

- Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated.  
Authentication can happen via asymmetric cryptography (e.g., RSA [[RSA](#)], the Elliptic Curve Digital Signature Algorithm (ECDSA) [[ECDSA](#)], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [[RFC8032](#)]) or a symmetric pre-shared key (PSK).
- Confidentiality: Data sent over the channel after establishment is only visible to the endpoints. TLS does not hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.
- Integrity: Data sent over the channel after establishment cannot be modified by attackers without detection.

First encryption

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [\[RFC5116\]](#), [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|  | <p>This specification defines the following cipher suites for use with TLS 1.3.</p> <table border="1"><thead><tr><th>Description</th><th>Value</th></tr></thead><tbody><tr><td>TLS_AES_128_GCM_SHA256</td><td>{0x13,0x01}</td></tr><tr><td>TLS_AES_256_GCM_SHA384</td><td>{0x13,0x02}</td></tr><tr><td>TLS_CHACHA20_POLY1305_SHA256</td><td>{0x13,0x03}</td></tr><tr><td>TLS_AES_128_CCM_SHA256</td><td>{0x13,0x04}</td></tr><tr><td>TLS_AES_128_CCM_8_SHA256</td><td>{0x13,0x05}</td></tr></tbody></table> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> | Description | Value | TLS_AES_128_GCM_SHA256 | {0x13,0x01} | TLS_AES_256_GCM_SHA384 | {0x13,0x02} | TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} | TLS_AES_128_CCM_SHA256 | {0x13,0x04} | TLS_AES_128_CCM_8_SHA256 | {0x13,0x05} |
|--|--|-------------|-------|------------------------|-------------|------------------------|-------------|------------------------------|-------------|------------------------|-------------|--------------------------|-------------|
| Description  | Value  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_128_GCM_SHA256                                     | {0x13,0x01}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_256_GCM_SHA384                                     | {0x13,0x02}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_CHACHA20_POLY1305_SHA256                               | {0x13,0x03}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_128_CCM_SHA256                                     | {0x13,0x04}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| TLS_AES_128_CCM_8_SHA256                                   | {0x13,0x05}  |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |
| encrypting a bit stream with a first encryption algorithm; | <p>The standard practices encrypting a bitstream (e.g., bitstream of digital certificate) with a first encryption algorithm (e.g., signature encryption algorithm i.e., SHA256RSA encryption algorithm).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA encryption algorithm) and generates a ciphertext.</p>   |             |       |                        |             |                        |             |                              |             |                        |             |                          |             |

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

As shown below, the accused instrumentality discloses the signature encryption algorithm.

The screenshot shows a Fiddler network traffic capture. On the left, a list of requests is displayed, showing various HTTP and HTTPS connections. Request 23 is highlighted with a red box and labeled "Tunnel to www.higginbotham.com:443". The right side of the interface shows a detailed view of this session. At the top of this view, the secure protocol is identified as "Secure Protocol: TLS 1.3" and the cipher suite as "Cipher Suite: TLS\_AES\_256\_GCM\_SHA384". Below this, the server certificate information is provided, including the subject (CN=higginbotham.com, Simple Name: higginbotham.com, DNS Name: higginbotham.com), version (V3), and issuer (CN=GTS CA 1P5, O=Google Trust Services LLC, C=US, Simple Name: GTS CA 1P5, DNS Name: GTS CA 1P5).

*Source: Fiddler Capture*

|  |   |
|--|---|
|  | <pre> SignedCertTimestamp (RFC5962) empty ALPN h2, http/1.1 signature_algs ecdsa_secp256r1_sha256,rsa_pss_rsae_sha256,rsa_pkcs1_sha256,ecdsa_secp384r1_sha384,rsa_pss_rsae_sha384,rsa_pkcs1_sha384,rsa_pss_rsae_sha512,rsa_pkcs1_sha512 0x001b 02 00 02 supported_groups grease [0x5a5a], unknown [0x6399], x25519 [0xd1], secp256r1 [0x17], secp384r1 [0x18] key_share 04 ED 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 39 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85 7A 95 44 34 07 19 C8 33 3F EC 9C 45 46 08 84 C9 1C 22 E1 3C 9F 87 J1 A3 5B C3 72 80 E1 C9 D5 CA 9D C4 69 B0 05 5B 9D E2 44 2B 56 F5 BF 82 84 67 9C C6 65 5F 82 3E B2 F0 B8 F5 3B AF 0F 86 11 A4 1C 00 5A 46 68 D9 57 AF 31 C6 C2 F4 F0 38 F1 EC 97 6A 21 78 C4 A8 AA B4 85 CE C9 85 67 D7 7A 0C EB AB 79 1D 83 3A B2 3B 60 D6 A3 C2 01 87 A7 F7 CC B8 A0 E5 1E 1F 55 88 34 65 2B BA CA 06 9C 3B 17 9D A6 11 B2 39 9F 34 00 6C CB D0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E 2F C0 61 1D A0 54 55 73 A1 99 23 8C 39 BA A0 C6 73 TA 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A 85 2C 21 80 CA 1F E2 AC B4 73 A5 FD EB 7B 6D 85 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F E0 75 BA 39 AA 0B 66 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F1 11 99 AD 04 48 DA 37 3B 2A F2 43 9B 66 23 56 C7 5A E0 5E 74 33 2F 3A 86 10 07 F7 D3 95 81 38 23 05 30 1B 5E A5 C7 07 13 84 84 A9 90 B3 66 5B 11 A9 A6 71 32 3E 15 DC EC 94 3E 55 93 83 64 C0 BF 79 2F C3 22 6C B1 C4 AB 5B 12 23 C1 C3 B1 E8 05 13 61 C7 42 DC 1C 3F F0 AB 47 A3 00 28 E6 A7 C1 C1 26 7B D7 FB 9F E9 09 14 5A E3 B4 5A 5B 9B D2 D5 47 0D 6B 7B 4A 71 6D 7E 01 37 F8 E0 A3 1E 58 6B D6 D2 C0 26 53 8B 9B 00 09 DF 5A 20 22 E9 A3 DD 00 9E 98 F0 4C A0 83 A0 B0 8F 26 98 2F 47 47 7B 2B 10 49 20 5F 14 53 0B 4E DA 90 E3 A0 82 98 C8 89 6E 63 TA 7A 8B AA A7 E6 C8 08 2D 4B 10 8A 70 A0 3A 93 D1 75 07 59 1C 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 99 3C E8 A3 A7 F0 73 D4 FA 81 9F 99 BF 70 B7 A1 renegotiation_info 00 0xfe0d 00 00 01 00 01 BB 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15 F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 62 57 A0 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E 2D 8A 8B BD F9 05 06 D4 63 E1 FC 35 82 4F 90 98 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 FB 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 04 20 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 89 5A server_name www.higginbotham.com orase (0x3a3a) 00 </pre> <p><b>Digital certificate</b></p> <p><b>First encryption algorithm</b></p> <p><b>Source: Fiddler Capture</b></p> |
|--|---|

The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

All handshake messages after the ServerHello are now encrypted. The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

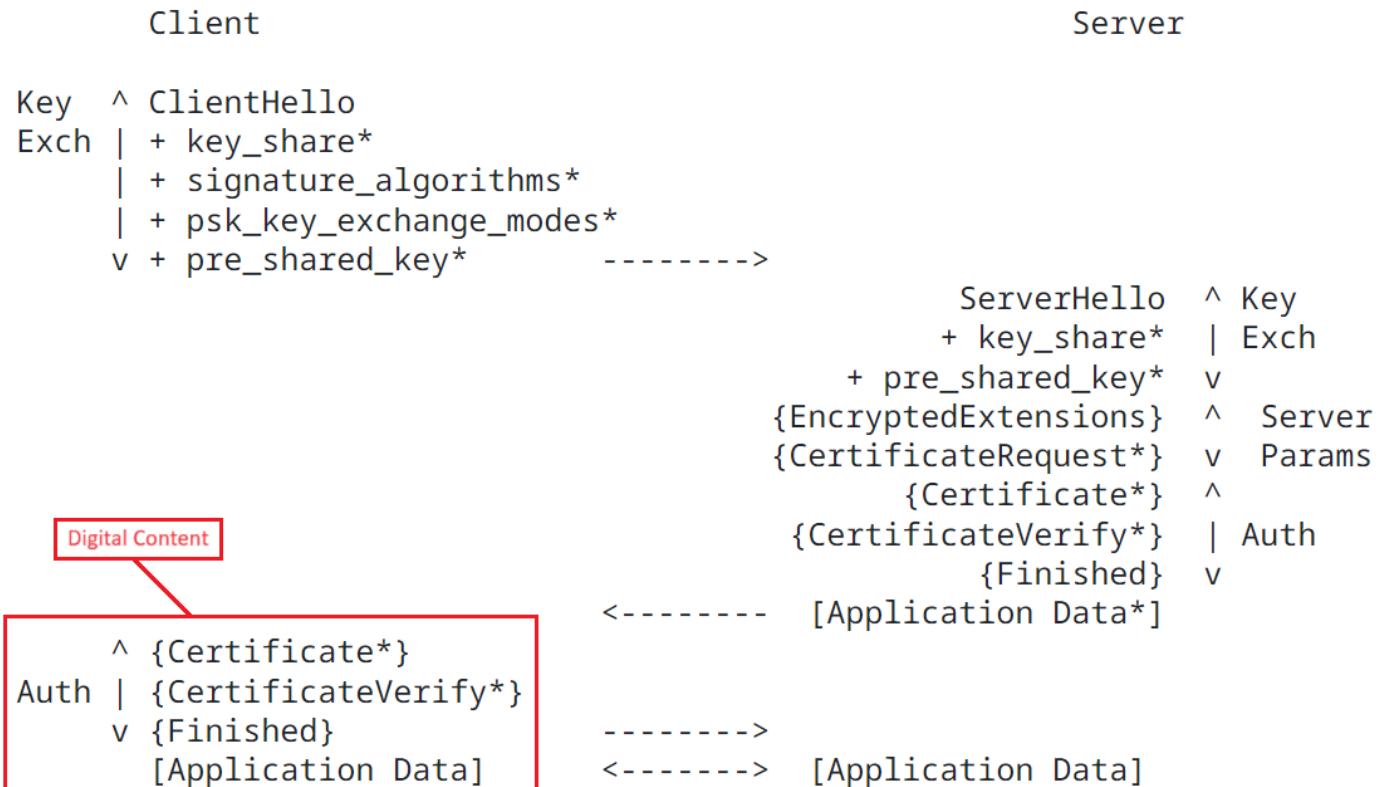
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.  
First encryption
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

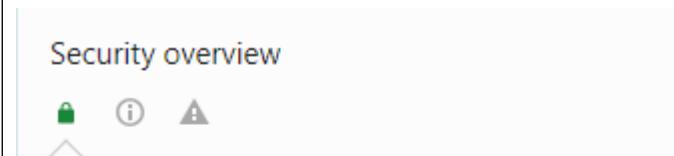
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |   |
|---|---|
|   | <p><b>Introduction</b></p> <p>The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream. Specifically, the secure channel should provide the following properties:</p> <ul style="list-style-type: none"><li>- Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated.<br/>Authentication can happen via asymmetric cryptography (e.g., RSA [RSA], the Elliptic Curve Digital Signature Algorithm (ECDSA) [ECDSA], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032]) or a symmetric pre-shared key (PSK).</li><li>- Confidentiality: Data sent over the channel after establishment is only visible to the endpoints. TLS does not hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.</li><li>- Integrity: Data sent over the channel after establishment cannot be modified by attackers without detection.</li></ul> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446">https://datatracker.ietf.org/doc/html/rfc8446</a></p> |
| associating a first decryption algorithm with the encrypted bit stream; | The standard practices associating a first decryption algorithm (e.g., signature decryption algorithm i.e., SHA256RSA decryption algorithm) with the encrypted bit stream (e.g., encrypted certificate with signature encryption algorithm).<br>The standard practices providing a two-level encryption security for data   |

communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA encryption algorithm) and generates a ciphertext.

The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it. The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate.

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

### The Transport Layer Security (TLS) Protocol Version 1.3

#### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

As shown below, the accused instrumentality discloses the signature decryption algorithm.

[Thumbprint]  
129157F7A5852AD23DF17FE2934E6396DB6B0251

[Signature Algorithm]  
sha256RSA(1.2.840.113549.1.1.11)

[Public Key]  
Algorithm: RSA  
Length: 2048

Key Blob: 30 82 01 0a 02 82 01 01 00 ca b8 66 5f 8a 37 94 14 e1 2f e3 49 cf d7 ad cb 20 8c 54 93 6b af 66 56 02 11 26 db 20 13 00 a0 ce fe ae e9 ac 8d 68 8f a0 56 fd ff b4 b1 4b 9c 54 ff 55 4a 95 76 b9 db 3c 77 5e 9a b1 dd 30 74 36 77 cb 92 12 df 17 62 6d 2a aa 74 3c f6 68 c8 34 8d eb 69 97 9e f2 43 02 fa d7 76 99 c0 bc b3 56 2d 17 ca 39 c8 00 5d 91 c8 1d 9b c9 70 c8 c1 b0 da 40 23 3e 9a fe a9 ed e4 60 ce 15 b0 af 43 a3 07 a5 81 25 dd c3 92 e7 72 9e 27 df bd 01 05 2a b4 60 b8 83 6b fa bc db 97 51 47 a8 2e d0 52 f7 c2 0a 1c 02 96 d2 c7 fd 04 a5 7e 78 04 ee 59 ed 33 9a e4 cc 42 bb 87 36 95 ac 4b 9b 6b 4f 96 4c ed 20 f7 bd 71 4b bc d8 91 81 3c be 56 c8 05 a6 e1 f7 66 85 88 46 91 96 a3 e4 e8 bb 45 f5 b8 78 25 4b 02 cb 23 e6 54 94 ed 94 f6 87 3d 29 55 31 eb 59 e9 a1 54 4c 50 b0 74 2c a7 b6 1e f7 23 02 03 01 00 01

Parameters: 05 00

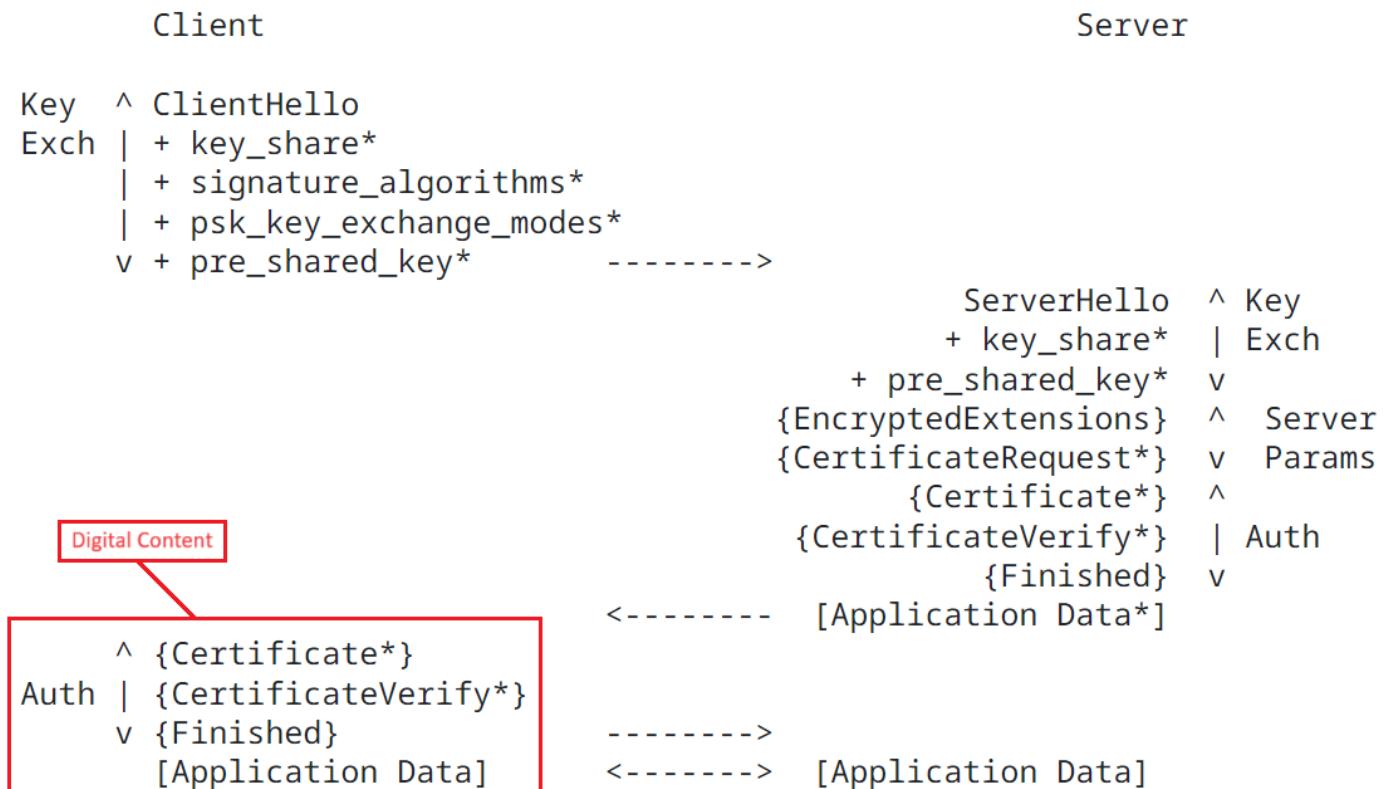
[Extensions]  
\* Key Usage(2.5.29.15):  
Digital Signature, Key Encipherment (a0)

First decryption algorithm

Source: Fiddler Capture

| OID description     |  |
|---------------------|--|
|                     | <div style="border: 1px solid red; padding: 5px; display: inline-block;">First decryption algorithm identifier</div>   |
| <b>OID:</b>         | {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1)}<br>sha256WithRSAEncryption(11)<br>1.2.840.113549.1.1.11<br>/ISO/Member-Body/US/113549/1/1/11  |
|                     | (ASN.1 notation)<br>(dot notation)<br>(OID-IRI notation)   |
| <b>Description:</b> | Public-Key Cryptography Standards (PKCS) #1 version 1.5 signature algorithm with Secure Hash Algorithm 256 (SHA256) and Rivest, Shamir and Adleman (RSA) encryption  |
|                     | <a href="http://oid-info.com/get/1.2.840.113549.1.1.11">http://oid-info.com/get/1.2.840.113549.1.1.11</a><br><br>-- When the following OIDs are used in an AlgorithmIdentifier, the<br>-- parameters MUST be present and MUST be NULL.<br><br>sha224WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 14 }<br><br><u>sha256WithRSAEncryption</u> OBJECT IDENTIFIER ::= { pkcs-1 11 }<br><br>sha384WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 12 }<br><br>sha512WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 13 }<br><a href="https://www.ietf.org/rfc/rfc4055.txt">https://www.ietf.org/rfc/rfc4055.txt</a> |

Figure 1 below shows the basic full TLS handshake:



#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.  
Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

All handshake messages after the ServerHello are now encrypted. The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

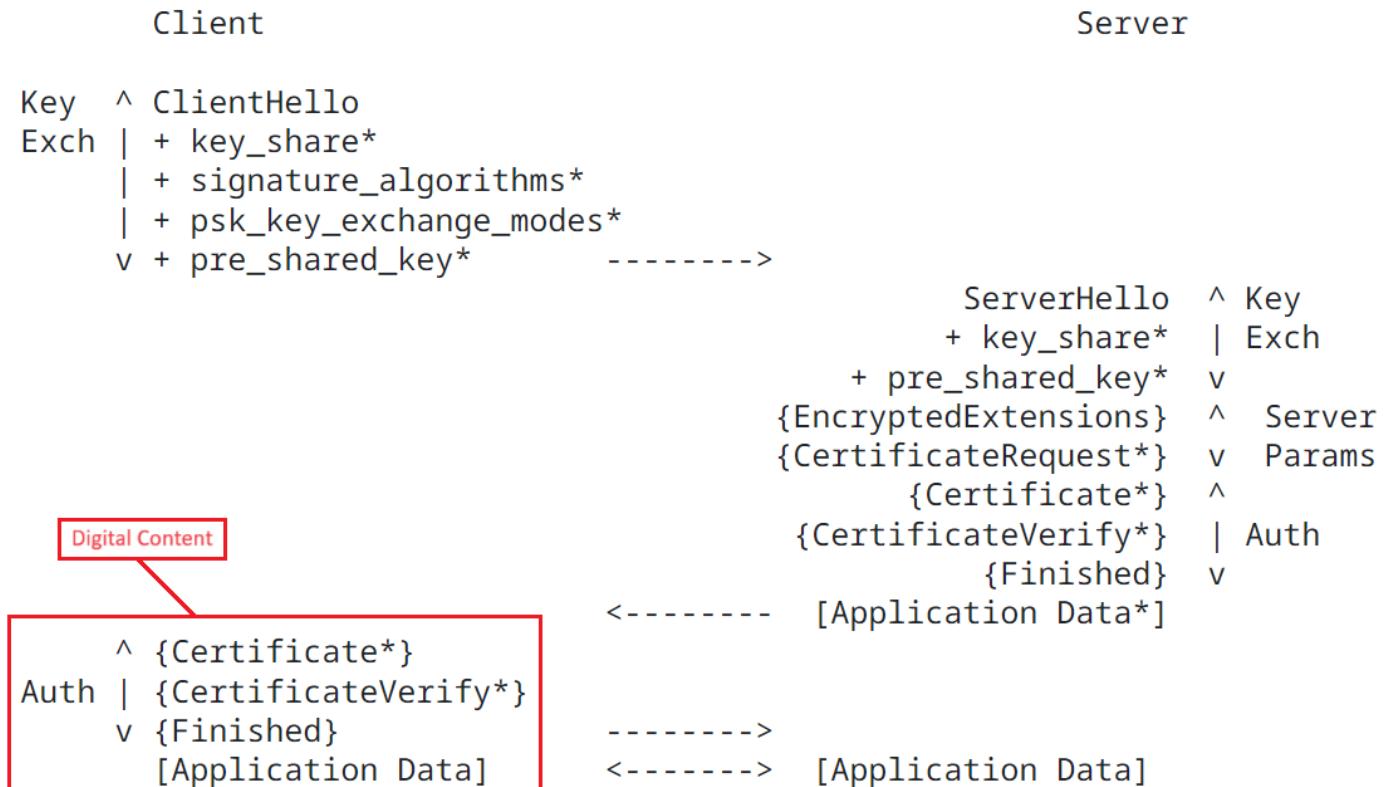
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.  
First encryption
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.3. Certificate Verify

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

First  
decryption  
algorithm  
information

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

As shown below, the receiving party will be able to decrypt the encrypted message with the provided signature decryption algorithm information i.e., SHA-256 RSA decryption algorithm.

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

|  |  |
|--|--|
|  | <p>The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A <i>cryptographic hash function</i> is a function that computes a <i>message authentication code</i> from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that <math>H</math> is a cryptographic hash function. To sign a message <math>m</math>, party A computes <math>h = D_A(H(m))</math> and sends <math>E_B(m, h)</math> to B. Party B now has evidence that A signed <math>m</math> because <math>E_A(h) = H(m)</math>, and A is the only one who could have generated a value <math>h</math> with that property.</p> |
| encrypting both the encrypted bit stream and the first decryption algorithm with a second encryption algorithm to yield a second bit stream; | <p>The standard practices encrypting both the encrypted bit stream (e.g., encrypted digital certificate) and the first decryption algorithm (e.g., signature decryption algorithm) with a second encryption algorithm (e.g., cipher suit selected from one of the AEAD algorithms such as TLS_AES_256_GCM_SHA384, etc.) to yield a second bit stream (e.g., TLS ciphertext bitstream).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA algorithm) and generates a ciphertext.</p> <p>The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it.</p>                           |

The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate. The standard further practices encrypting the authentication message, including the encrypted digital certification and the certificate verify message, with a second decryption algorithm i.e., AEAD algorithm such as TLS\_AES\_256\_GCM\_SHA384, etc.

#### Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.

[View certificate](#)

- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.

- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

| Headers   | TextView   | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm |
|-----------|--|------------|----------|---------|------|---------|-----|------|-----|-----------------------------|
| 00000019  | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 |            |          |         |      |         |     |      |     |                             |
| 00000032  | 77 77 2E 68 69 67 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A |            |          |         |      |         |     |      |     |                             |
| 0000004B  | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55 |            |          |         |      |         |     |      |     |                             |
| 00000064  | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 |            |          |         |      |         |     |      |     |                             |
| 0000007D  | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36 |            |          |         |      |         |     |      |     |                             |
| 00000096  | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48 |            |          |         |      |         |     |      |     |                             |
| 000000AF  | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31 |            |          |         |      |         |     |      |     |                             |
| 000000C8  | 32 36 2E 30 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D |            |          |         |      |         |     |      |     |                             |
| 000000E1  | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E |            |          |         |      |         |     |      |     |                             |
| 000000FA  | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E |            |          |         |      |         |     |      |     |                             |
| 000000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20 |            |          |         |      |         |     |      |     |                             |
| 00000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65 |            |          |         |      |         |     |      |     |                             |
| 00000145  | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72 |            |          |         |      |         |     |      |     |                             |
| 0000015E  | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53 |            |          |         |      |         |     |      |     |                             |
| 00000177  | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69 |            |          |         |      |         |     |      |     |                             |
| 00000190  | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A |            |          |         |      |         |     |      |     |                             |
| 000001A9  | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20 |            |          |         |      |         |     |      |     |                             |
| 000001C2  | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30 |            |          |         |      |         |     |      |     |                             |
| 000001DB  | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37 |            |          |         |      |         |     |      |     |                             |
| 000001F4  | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69 |            |          |         |      |         |     |      |     |                             |
| 0000020D  | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A |            |          |         |      |         |     |      |     |                             |
| 00000226  | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32 |            |          |         |      |         |     |      |     |                             |
| 0000023F  | 20 34 44 20 32 43 20 43 42 20 36 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |            |          |         |      |         |     |      |     |                             |

Source: Fiddler Capture

|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | OC 2A 73 3E B4 85 7A 95   |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     |                           |

*Source: Fiddler Capture*

The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm. The authentication message is a TLS plaintext.

handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [\[RFC5116\]](#), [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

All handshake messages after the ServerHello are now encrypted.  
The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

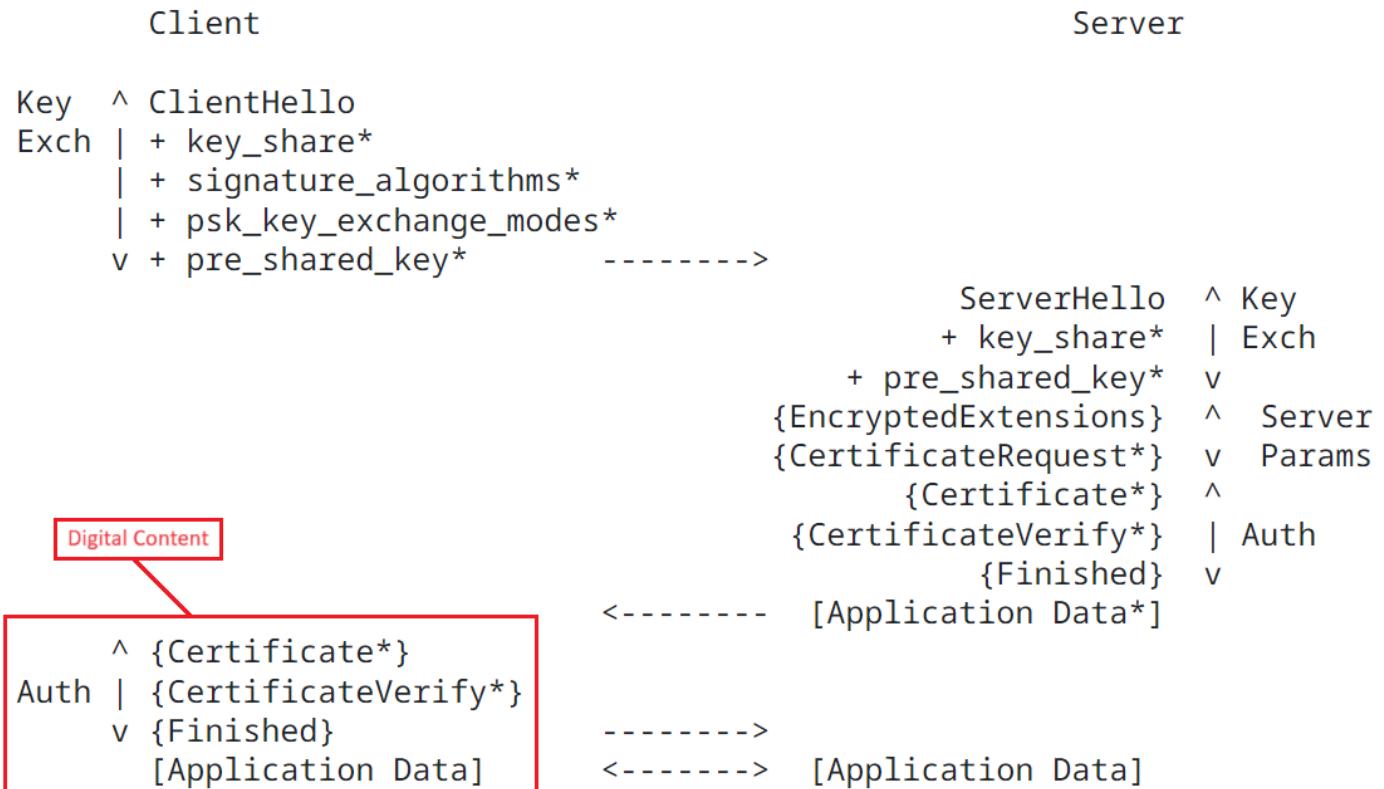
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4. Authentication Messages

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block. These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.1.1. Cryptographic Negotiation

In TLS, the cryptographic negotiation proceeds by the client offering the following four sets of options in its ClientHello:

- A list of cipher suites which indicates the AEAD algorithm/HKDF hash pairs which the client supports.
- A "supported\_groups" ([Section 4.2.7](#)) extension which indicates the (EC)DHE groups which the client supports and a "key\_share" ([Section 4.2.8](#)) extension which contains (EC)DHE shares for some or all of these groups.
- A "signature\_algorithms" ([Section 4.2.3](#)) extension which indicates the signature algorithms which the client can accept. A "signature\_algorithms\_cert" extension ([Section 4.2.3](#)) may also be added to indicate certificate-specific signature algorithms.
- A "pre\_shared\_key" ([Section 4.2.11](#)) extension which contains a list of symmetric key identities known to the client and a "psk\_key\_exchange\_modes" ([Section 4.2.9](#)) extension which indicates the key exchange modes that may be used with PSKs.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.3. Certificate Verify

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

First  
decryption  
algorithm  
information

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |   |
|---|---|
|   | <p>The <u>"extension_data"</u> field of these extensions contains a <u>SignatureSchemeList</u> value:</p> <pre> enum {     /* RSASSA-PKCS1-v1_5 algorithms */     rsa_pkcs1_sha256(0x0401),     rsa_pkcs1_sha384(0x0501),     rsa_pkcs1_sha512(0x0601),      /* ECDSA algorithms */     ecdsa_secp256r1_sha256(0x0403),     ecdsa_secp384r1_sha384(0x0503),     ecdsa_secp521r1_sha512(0x0603),      /* RSASSA-PSS algorithms with public key OID rsaEncryption */     rsa_pss_rsae_sha256(0x0804),     rsa_pss_rsae_sha384(0x0805),     rsa_pss_rsae_sha512(0x0806),      /* EdDSA algorithms */     ed25519(0x0807),     ed448(0x0808),      /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */     rsa_pss_pss_sha256(0x0809),     rsa_pss_pss_sha384(0x080a),     rsa_pss_pss_sha512(0x080b), </pre> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| associating a second decryption algorithm with the second bit stream. | <p>The standard practices associating a second decryption algorithm (e.g., cipher suit selected from one of the AEAD algorithms such as TLS_AES_256_GCM_SHA384, etc.) with the second bit stream (e.g., TLS ciphertext bitstream).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA algorithm) and generates a ciphertext.</p>   |

The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it. The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate. The standard further practices encrypting the authentication message, including the encrypted digital certification and the certificate verify message, with a second decryption algorithm i.e., AEAD algorithm such as TLS\_AES\_256\_GCM\_SHA384, etc.

#### Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.

[View certificate](#)

- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.

- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

The screenshot shows a Fiddler network traffic capture. On the left, a list of requests and responses is displayed, mostly in green (HTTP) and red (HTTPS). A specific entry for a CONNECT tunnel to `www.higginbotham.com:443` is highlighted with a red box. On the right, the Fiddler interface displays detailed session information for this connection. It shows the secure protocol as `TLS 1.3` and the cipher suite as `TLS_AES_256_GCM_SHA384`. The server certificate is analyzed, showing the version as V3, the subject as `CN=higginbotham.com`, and the issuer as `CN=GTS CA 1P5, O=Google Trust Services LLC, C=US`. The DNS name is listed as `higginbotham.com`.

*Source: Fiddler Capture*

|          | <p>Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.</p> <p>Secure Protocol: TLS 1.3<br/>Cipher Suite: TLS_AES_256_GCM_SHA384</p> <pre>-- Server Certificate ----- [Version] V3  [Subject] CN=higginbotham.com Simple Name: higginbotham.com DNS Name: higginbotham.com  [Issuer] CN=GTS CA 1P5, O=Google Trust Services LLC, C=US</pre> <p><i>Source: Fiddler Capture</i></p>   |            |          |            |          |         |      |         |     |                           |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
|----------|--|------------|----------|------------|----------|---------|------|---------|-----|---------------------------|-----|------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|--------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|-------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|----------|--|--|--|--|--|--|--|--|--|---------------------------|
|          | <table border="1"> <thead> <tr> <th>Headers</th><th>TextView</th><th>SyntaxView</th><th>WebForms</th><th>HexView</th><th>Auth</th><th>Cookies</th><th>Raw</th><th>JSON</th><th>XML</th><th>Second bitstream</th></tr> </thead> <tbody> <tr><td>000004C9</td><td>36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>67 98 9F 8B 88 D9 A0 C6 1</td></tr> <tr><td>000004E2</td><td>35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>5 CC 37 BB 51 91 1D 30 CE</td></tr> <tr><td>000004FB</td><td>20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>8D F6 8E 5F 54 C2 5A E3</td></tr> <tr><td>00000514</td><td>38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>9 2C 7B 89 AA BB C5 AF E</td></tr> <tr><td>0000052D</td><td>39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>9 56 10 B8 15 57 94 A1 A1</td></tr> <tr><td>00000546</td><td>20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>0C 2A 73 3E B4 85 7A 95</td></tr> <tr><td>0000055F</td><td>34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>44 34 07 19 C8 33 3F EC 9</td></tr> <tr><td>00000578</td><td>43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>C 45 46 0B 84 C9 1C 22 E1</td></tr> <tr><td>00000591</td><td>20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>3C 9F 87 71 A3 5B C3 72</td></tr> <tr><td>000005AA</td><td>42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>B0 E1 C9 D5 CA 9D C4 69 B</td></tr> <tr><td>000005C3</td><td>30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>0 05 5B 9D E2 A4 2B 56 F5</td></tr> <tr><td>000005DC</td><td>20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>BF 82 84 67 9C C6 65 5F</td></tr> <tr><td>000005F5</td><td>38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>82 3E B2 F0 BB F5 3B AF 0</td></tr> <tr><td>0000060E</td><td>46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>F 86 11 A4 1C 00 5A 46 69</td></tr> <tr><td>00000627</td><td>20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>D9 57 AF 31 C6 C2 F4 F0</td></tr> <tr><td>00000640</td><td>33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>38 F7 EC 97 6A 21 78 C4 A</td></tr> <tr><td>00000659</td><td>38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>8 AA B4 85 CE C9 85 67 D7</td></tr> <tr><td>00000672</td><td>20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>7A 0C EB AB 79 1D 83 3A</td></tr> <tr><td>000006BB</td><td>42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>B2 3B 60 D6 A3 C2 01 87 A</td></tr> <tr><td>000006A4</td><td>37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>7 F7 CC B8 A0 E5 1E 1F 55</td></tr> <tr><td>000006BD</td><td>20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>88 34 65 2F BA CA 06 9C</td></tr> <tr><td>000006D6</td><td>33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>3B 17 9D A6 11 B2 39 9F 3</td></tr> <tr><td>000006EF</td><td>34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>4 00 6C CB D0 53 3E 57 16</td></tr> </tbody> </table> <p><i>Source: Fiddler Capture</i></p> <p>The standard defines four record message types, including a handshake message type. The handshake messages are communicated to establish a secure channel for TLS</p> | Headers    | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON                      | XML | Second bitstream | 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |  |  |  |  |  |  |  |  | 67 98 9F 8B 88 D9 A0 C6 1 | 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |  |  |  |  |  |  |  |  | 5 CC 37 BB 51 91 1D 30 CE | 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |  |  |  |  |  |  |  |  | 8D F6 8E 5F 54 C2 5A E3 | 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |  |  |  |  |  |  |  |  | 9 2C 7B 89 AA BB C5 AF E | 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |  |  |  |  |  |  |  |  | 9 56 10 B8 15 57 94 A1 A1 | 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |  |  |  |  |  |  |  |  | 0C 2A 73 3E B4 85 7A 95 | 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |  |  |  |  |  |  |  |  | 44 34 07 19 C8 33 3F EC 9 | 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |  |  |  |  |  |  |  |  | C 45 46 0B 84 C9 1C 22 E1 | 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |  |  |  |  |  |  |  |  | 3C 9F 87 71 A3 5B C3 72 | 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |  |  |  |  |  |  |  |  | B0 E1 C9 D5 CA 9D C4 69 B | 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |  |  |  |  |  |  |  |  | 0 05 5B 9D E2 A4 2B 56 F5 | 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |  |  |  |  |  |  |  |  | BF 82 84 67 9C C6 65 5F | 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |  |  |  |  |  |  |  |  | 82 3E B2 F0 BB F5 3B AF 0 | 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |  |  |  |  |  |  |  |  | F 86 11 A4 1C 00 5A 46 69 | 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |  |  |  |  |  |  |  |  | D9 57 AF 31 C6 C2 F4 F0 | 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |  |  |  |  |  |  |  |  | 38 F7 EC 97 6A 21 78 C4 A | 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |  |  |  |  |  |  |  |  | 8 AA B4 85 CE C9 85 67 D7 | 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |  |  |  |  |  |  |  |  | 7A 0C EB AB 79 1D 83 3A | 000006BB | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |  |  |  |  |  |  |  |  | B2 3B 60 D6 A3 C2 01 87 A | 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |  |  |  |  |  |  |  |  | 7 F7 CC B8 A0 E5 1E 1F 55 | 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |  |  |  |  |  |  |  |  | 88 34 65 2F BA CA 06 9C | 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |  |  |  |  |  |  |  |  | 3B 17 9D A6 11 B2 39 9F 3 | 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |  |  |  |  |  |  |  |  | 4 00 6C CB D0 53 3E 57 16 |
| Headers  | TextView   | SyntaxView | WebForms | HexView    | Auth     | Cookies | Raw  | JSON    | XML | Second bitstream          |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31   |            |          |            |          |         |      |         |     | 67 98 9F 8B 88 D9 A0 C6 1 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45   |            |          |            |          |         |      |         |     | 5 CC 37 BB 51 91 1D 30 CE |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20   |            |          |            |          |         |      |         |     | 8D F6 8E 5F 54 C2 5A E3   |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45   |            |          |            |          |         |      |         |     | 9 2C 7B 89 AA BB C5 AF E  |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31   |            |          |            |          |         |      |         |     | 9 56 10 B8 15 57 94 A1 A1 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20   |            |          |            |          |         |      |         |     | 0C 2A 73 3E B4 85 7A 95   |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39   |            |          |            |          |         |      |         |     | 44 34 07 19 C8 33 3F EC 9 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31   |            |          |            |          |         |      |         |     | C 45 46 0B 84 C9 1C 22 E1 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20   |            |          |            |          |         |      |         |     | 3C 9F 87 71 A3 5B C3 72   |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42   |            |          |            |          |         |      |         |     | B0 E1 C9 D5 CA 9D C4 69 B |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35   |            |          |            |          |         |      |         |     | 0 05 5B 9D E2 A4 2B 56 F5 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20   |            |          |            |          |         |      |         |     | BF 82 84 67 9C C6 65 5F   |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30   |            |          |            |          |         |      |         |     | 82 3E B2 F0 BB F5 3B AF 0 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39   |            |          |            |          |         |      |         |     | F 86 11 A4 1C 00 5A 46 69 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20   |            |          |            |          |         |      |         |     | D9 57 AF 31 C6 C2 F4 F0   |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41   |            |          |            |          |         |      |         |     | 38 F7 EC 97 6A 21 78 C4 A |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37   |            |          |            |          |         |      |         |     | 8 AA B4 85 CE C9 85 67 D7 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20   |            |          |            |          |         |      |         |     | 7A 0C EB AB 79 1D 83 3A   |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006BB | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41   |            |          |            |          |         |      |         |     | B2 3B 60 D6 A3 C2 01 87 A |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35   |            |          |            |          |         |      |         |     | 7 F7 CC B8 A0 E5 1E 1F 55 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20   |            |          |            |          |         |      |         |     | 88 34 65 2F BA CA 06 9C   |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33   |            |          |            |          |         |      |         |     | 3B 17 9D A6 11 B2 39 9F 3 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36   |            |          |            |          |         |      |         |     | 4 00 6C CB D0 53 3E 57 16 |     |                  |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                          |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                         |          |  |  |  |  |  |  |  |  |  |                           |          |  |  |  |  |  |  |  |  |  |                           |

communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

Further, the AEAD encrypted message comprises a ciphertext (e.g., encrypted ciphertext after the encryption by the second encryption algorithm), nonce (e.g., associating second decryption algo), key and associated data. The maximum length of nonce is a cipher suit specific element. The nonce and associated data are utilized in decryption of the AEAD encrypted message.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.2. Authenticated Decryption

Second decryption algorithm

The authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic. A ciphertext C, a nonce N, and associated data A are authentic for key K when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [RFC5116].. [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:  
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

All handshake messages after the ServerHello are now encrypted.  
The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

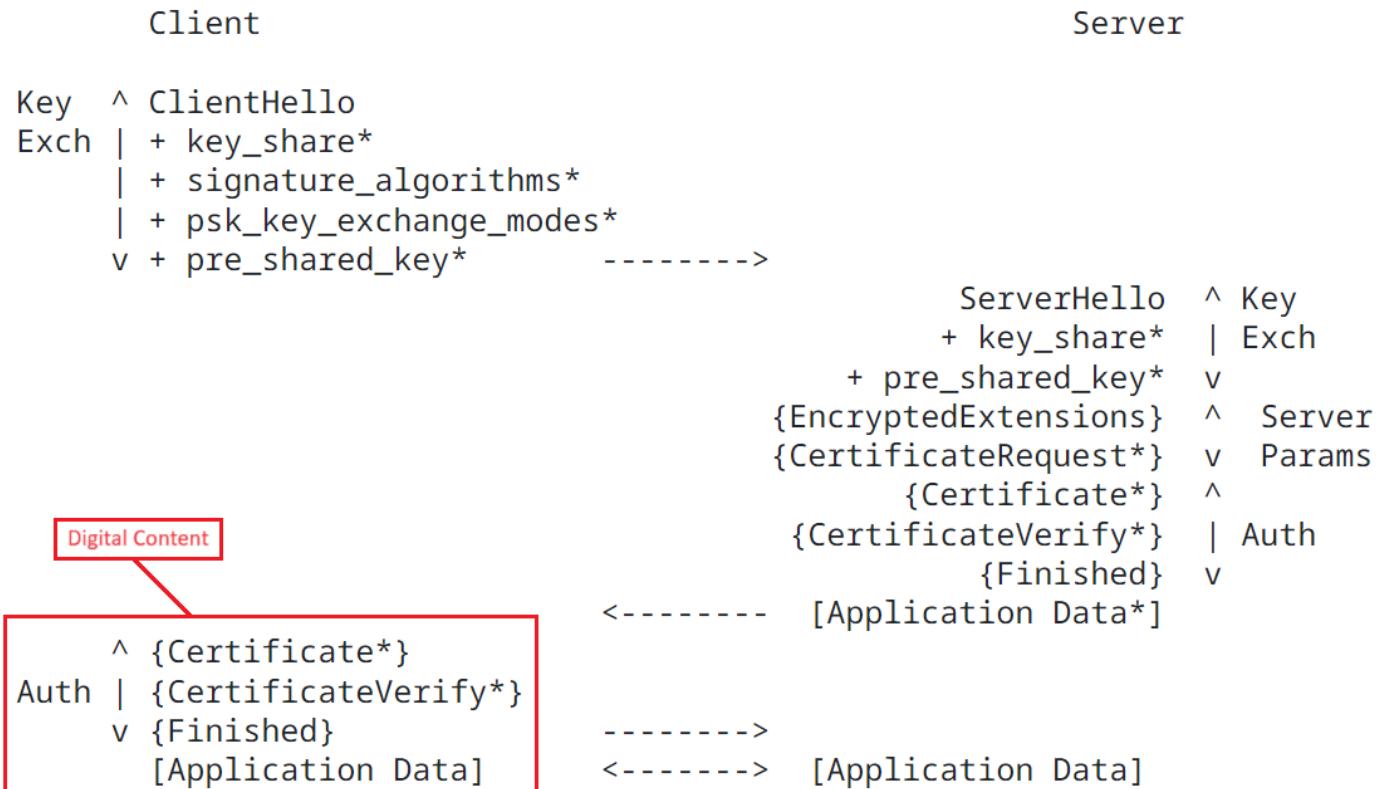
#### **4.4. Authentication Messages**

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block.

These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|   |  |
|---|--|
|   | <p>The <u>"extension_data"</u> field of these extensions contains a <u>SignatureSchemeList</u> value:</p> <pre>enum {     /* RSASSA-PKCS1-v1_5 algorithms */     rsa_pkcs1_sha256(0x0401),     rsa_pkcs1_sha384(0x0501),     rsa_pkcs1_sha512(0x0601),      /* ECDSA algorithms */     ecdsa_secp256r1_sha256(0x0403),     ecdsa_secp384r1_sha384(0x0503),     ecdsa_secp521r1_sha512(0x0603),      /* RSASSA-PSS algorithms with public key OID rsaEncryption */     rsa_pss_rsae_sha256(0x0804),     rsa_pss_rsae_sha384(0x0805),     rsa_pss_rsae_sha512(0x0806),      /* EdDSA algorithms */     ed25519(0x0807),     ed448(0x0808),      /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */     rsa_pss_pss_sha256(0x0809),     rsa_pss_pss_sha384(0x080a),     rsa_pss_pss_sha512(0x080b), <a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></pre> |
| 38. The software system or computer program of claim 37, further translatable for | The standard further discloses decrypting the first bit stream (e.g., encrypted digital certificate with signature encryption algorithm i.e., SHA-256 RSA, etc.) and the second bit stream (e.g., a second-level encryption with AEAD encryption algorithm such as TLS AES 256 GCM SHA384, etc.) with the first associated decryption  |

|  |   |
|--|---|
| decrypting the first bit stream and the second bit stream with the first associated decryption algorithm and the second associated decryption algorithm wherein the decryption is accomplished by a target unit. | <p>algorithm (e.g., signature decryption algorithm i.e., SHA-256 RSA, etc.) and the second associated decryption algorithm (e.g., cipher suit selected from one of the AEAD decryption algorithms such as TLS_AES_256_GCM_SHA384, etc.) wherein the decryption is accomplished by a target unit (e.g., a server of the accused instrumentality).</p> <p>The standard practices providing a two-level encryption security for data communication. It encrypts a plaintext with a first encryption technique i.e., signature encryption algorithm (e.g., SHA256RSA algorithm) and generates a ciphertext.</p> <p>The standard defines an authentication message, communicated after the hello handshake messages, which comprises an encrypted digital certificate with the signature encryption algorithm and an associated certificate verify message with it. The certificate verify message includes a signature algorithm extension field which provides information for the decryption of the encrypted digital certificate. The standard further practices encrypting the authentication message, including the encrypted digital certification and the certificate verify message, with a second decryption algorithm i.e., AEAD algorithm such as TLS_AES_256_GCM_SHA384, etc.</p> |
|--|---|

Security overview



This page is secure (valid HTTPS).

- Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by GTS CA 1P5.  
[View certificate](#)
- Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519Kyber768Draft00, and AES\_128\_GCM.
- Resources - all served securely

All resources on this page are served securely.

<https://www.higginbotham.com/>

## The Transport Layer Security (TLS) Protocol Version 1.3

### Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

<https://datatracker.ietf.org/doc/html/rfc8446>

The screenshot shows a Fiddler network traffic capture. On the left, a list of requests is displayed, showing various HTTP and HTTPS connections between the user's browser and various websites like higginbotham.com, googleapis.com, and cdnjs.cloudflare.com. Request number 8 is highlighted with a red box and labeled "Tunnel to www.higginbotham.com:443". Request number 23 is also highlighted with a green box and labeled "Tunnel to www.higginbotham.com:443". On the right, a detailed SSL/TLS analysis is shown for the selected connection. It includes tabs for Transformer, Headers, TextView, SyntaxView, ImageView, HexView, WebView, Auth, Caching, Cookies, Raw, JSON, and XML. The Headers tab shows the secure protocol as "TLS 1.3" and the cipher suite as "TLS\_AES\_256\_GCM\_SHA384". The Transformer tab shows the server certificate details, including the subject (CN=higginbotham.com), version (V3), and issuer (CN=GTS CA 1P5, O=Google Trust Services LLC, C=US). The Headers tab also lists the simple name and DNS name as higginbotham.com.

*Source: Fiddler Capture*

|  |  |
|--|--|
|  | <pre> SignedCertTimestamp (RFC5962) empty ALPN h2, http/1.1 signature_algs ecdsa_secp256r1_sha256,rsa_pss_rsae_sha256,rsa_pkcs1_sha256,ecdsa_secp384r1_sha384,rsa_pss_rsae_sha384,rsa_pkcs1_sha384,rsa_pss_rsae_sha512,rsa_pkcs1_sha512 0x001b 02 00 02 supported_groups grease [0x5a5a], unknown [0x6399], x25519 [0xd1], secp256r1 [0x17], secp384r1 [0x18] key_share 04 ED 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 39 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85 7A 95 44 34 07 19 C8 33 3F EC 9C 45 46 08 84 C9 1C 22 E1 3C 9F 87 71 A3 5B C3 72 80 E1 C9 D5 CA 9D C4 69 B0 05 5B 9D E2 44 2B 56 F5 BF 82 84 67 9C C6 65 5F 82 3E B2 F0 B8 F5 3B AF 0F 86 11 A4 1C 00 5A 46 68 D9 57 AF 31 C6 C2 F4 F0 38 F1 EC 97 6A 21 78 C4 A8 AA B4 85 CE C9 85 67 D7 7A 0C EB AB 79 1D 83 3A B2 3B 60 D6 A3 C2 01 87 A7 F7 CC B8 A0 E5 1E 1F 55 88 34 65 2B BA CA 06 9C 3B 17 9D A6 11 B2 39 9F 34 00 6C CB D0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E 2F C0 61 1D A0 54 55 73 A1 99 23 8C 39 BA A0 C6 73 A8 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A 85 2C 21 80 CA 1F E2 AC B4 73 A5 FD EB 7B 6D 85 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F 0E 75 BA 39 AA 0B 66 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F1 11 99 AD 04 48 DA 37 3B 79 2A 0F 73 C7 5A E0 5E 74 33 2F 3A 86 10 07 F7 3D 95 81 38 70 23 05 30 1B 5E A5 C7 07 13 84 84 A9 90 B3 66 5B 11 A9 A6 71 2C 15 DC EC 94 3E 55 93 83 64 C0 BF 79 2F C3 22 6C B1 C4 AB 5B 12 23 C1 C3 B1 E8 05 13 61 C7 42 DC 1C 3F F0 AB 47 A3 00 28 E6 A7 C1 C1 26 7B D7 FB 9F E9 09 14 5A E3 B4 5A 5B 9B D2 D5 47 0D 6B 7B 4A 71 6D 7E 01 37 F8 E0 A3 1E 58 6B D6 D2 C0 26 53 B8 9B 00 09 DF 5A 20 22 E9 A3 DD 00 9E 98 F0 4C A0 83 A0 B0 8F 26 98 2F 47 47 7B 2B 10 49 20 8F 4E DA 90 E3 A0 82 98 C8 89 6E 63 7A 8A AA A7 E6 C8 08 2D 4B 10 8A 70 A0 3A 93 D1 75 07 59 1C 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 99 3C E8 A3 A7 F0 73 D4 FA 81 9F 99 BF 70 B7 A1 08 BB 58 A8 C2 2D C1 FB BB 27 89 CB F0 92 BB A8 65 B3 E4 6C 6B 59 20 8B 55 13 2B 49 EB A3 13 B3 78 A5 F1 3A 70 86 8B 8B 96 7F 23 9C 37 19 0C C0 3C 0B 14 8F 67 78 4A F5 03 04 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 A4 AB CA 6B 3C 4A 85 2D C1 FB BB 27 89 CB F0 92 BB A8 65 B3 E4 6C 6B 59 20 8B 55 13 2B 49 EB A3 13 B3 78 A5 F1 3A 70 86 8B 8B 96 7F 23 9C 37 19 0C C0 3C 0B 14 8F 67 78 4A F5 03 04 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 A4 AB CA 6B 3C 4A 85 2D C1 FB BB 27 89 CB F0 92 BB A8 65 B3 E4 6C 6B 59 20 8B 55 13 2B 49 EB A3 13 B3 78 A5 F1 3A 70 86 8B 8B 96 7F 23 9C 37 19 0C C0 3C 0B 14 8F 67 78 4A F5 03 04 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 A4 AB 7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24 extended_master_secret empty ec_point_format uncompressed [0x0] status_request OCSP- Implicit Responder psk_key_exchange_modes 01 01 renegotiation_info 00 0xfe0d 00 00 01 00 01 BB 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15 F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 62 57 A0 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E 2D 8A 8B BD F9 05 06 D4 63 E1 FC 35 82 4F 90 98 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 FB 90 B6 F6 B1 A4 44 A1 D9 62 79 85 26 3D 37 D4 80 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 89 5A server_name www.higginbotham.com orase (0x3a3a) 00 </pre> <p><b>First encryption algorithm</b></p> <p><b>Source: Fiddler Capture</b></p> |
|--|--|

[Thumbprint]  
129157F7A5852AD23DF17FE2934E6396DB6B0251

[Signature Algorithm]  
sha256RSA(1.2.840.113549.1.1.11)

First decryption  
algorithm

[Public Key]  
Algorithm: RSA  
Length: 2048

Key Blob: 30 82 01 0a 02 82 01 01 00 ca b8 66 5f 8a 37 94 14 e1 2f e3 49 cf d7 ad cb 20 8c 54 93 6b af 66 56 02 11 26 db 20 13 00 a0 ce fe ae e9 ac 8d 68 8f a0 56 fd ff b4 14 b9 c5 54 ff 55 4a 95 76 b9 db 3c 77 5e 9a b1 dd 30 74 36 77 cb 92 12 df 17 62 6d 2a aa 74 3c f6 68 c8 34 8d eb 69 97 9e f2 43 02 fa d7 76 99 c0 bc b3 56 2d 17 ca 39 c8 00 5d 91 c8 1d 9b c9 70 c8 c1 b0 da 40 23 3e 9a fe a9 ed e4 60 ce 15 b0 af 43 a3 07 a5 81 25 dd c3 92 e7 72 9e 27 df bd 01 05 2a b4 60 b8 83 6b fa bc db 97 51 47 a8 2e d0 52 7f c2 0a 1c 02 96 d2 c7 fd 04 a5 7e 78 04 ee 59 ed 33 9a e4 cc 42 bb 87 36 95 ac 4b 9b 6b 4f 96 4c ed 20 f7 bd 71 4b bc d8 91 81 3c be 56 c8 05 a6 e1 7f 66 85 88 46 91 96 a3 e4 e8 bb 45 f5 b8 78 25 4b 02 cb 23 e6 54 94 ed 94 f6 87 3d 29 55 31 eb 59 e9 a1 54 4c 50 b0 74 2c a7 b6 1e 7f 23 02 03 01 00 01  
Parameters: 05 00

[Extensions]  
\* Key Usage(2.5.29.15):  
Digital Signature, Key Encipherment (a0)

*Source: Fiddler Capture*

| Headers  | Text View  | Syntax View | Web Forms | Hex View | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm   |
|----------|--|-------------|-----------|----------|------|---------|-----|------|-----|---|
| 00000019 | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 |             |           |          |      |         |     |      |     | com:443 HTTP/1.1..Host: www.higginbotham.com:443..Connection: keep-alive..User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36...A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below...Secure |
| 00000032 | 77 77 2E 68 69 67 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A |             |           |          |      |         |     |      |     |   |
| 0000004B | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55 |             |           |          |      |         |     |      |     |   |
| 00000064 | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 |             |           |          |      |         |     |      |     |   |
| 0000007D | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36 |             |           |          |      |         |     |      |     |   |
| 00000096 | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48 |             |           |          |      |         |     |      |     |   |
| 000000AF | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31 |             |           |          |      |         |     |      |     |   |
| 000000CB | 32 36 2E 30 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D |             |           |          |      |         |     |      |     |   |
| 000000E1 | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E |             |           |          |      |         |     |      |     |   |
| 000000FA | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E |             |           |          |      |         |     |      |     |   |
| 00000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20 |             |           |          |      |         |     |      |     |   |
| 0000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65 |             |           |          |      |         |     |      |     |   |
| 00000145 | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72 |             |           |          |      |         |     |      |     |   |
| 0000015E | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53 |             |           |          |      |         |     |      |     |   |
| 00000177 | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69 |             |           |          |      |         |     |      |     |   |
| 00000190 | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A |             |           |          |      |         |     |      |     |   |
| 000001A9 | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20 |             |           |          |      |         |     |      |     |   |
| 000001C2 | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30 |             |           |          |      |         |     |      |     |   |
| 000001DB | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37 |             |           |          |      |         |     |      |     |   |
| 000001F4 | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69 |             |           |          |      |         |     |      |     |   |
| 0000020D | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A |             |           |          |      |         |     |      |     |   |
| 00000226 | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32 |             |           |          |      |         |     |      |     |   |
| 0000023F | 20 34 44 20 32 43 20 43 42 20 36 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |             |           |          |      |         |     |      |     |   |

*Source: Fiddler Capture*

|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 39 2C 7B 89 AA BB C5 AF E |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | 0C 2A 73 3E B4 85 7A 95   |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |

*Source: Fiddler Capture*

Encrypted HTTPS traffic flows through this CONNECT tunnel. HTTPS Decryption is enabled in Fiddler, so decrypted sessions running in this tunnel will be shown in the Web Sessions list.

Secure Protocol: TLS 1.3  
 Cipher Suite: TLS\_AES\_256\_GCM\_SHA384

-- Server Certificate -----  
 [Version]  
 V3

[Subject]  
 CN=higginbotham.com  
 Simple Name: higginbotham.com  
 DNS Name: higginbotham.com

[Issuer]  
 CN=GTS CA 1P5, O=Google Trust Services LLC, C=US

*Source: Fiddler Capture*

The standard defines four record message types, including a handshake message type.

The handshake messages are communicated to establish a secure channel for TLS communication. A client device and a server negotiate an AEAD algorithm for encrypting TLS record message data. It discloses that after Hello handshake messages i.e., a ClientHello message and a ServerHello message, all handshake messages are encrypted with the negotiated encryption algorithm. One of the handshake messages after hello handshake messages i.e., an authentication message from the client, comprises a digital certificate encrypted by a signature encryption algorithm and a certificate verify message comprising information related to the signature decryption algorithm.

As shown below, the digital certificate is encrypted with the signature encryption algorithm and the certificate verify message, associated with the encrypted digital certificate, has a signature algorithm extension field that provides information related to the signature decryption algorithm. The authentication message is a TLS plaintext handshake message. This message is again encrypted with the negotiated AEAD encryption algorithm, e.g., recursive security protocol. The AEAD encrypted message is communicated between the client and the server.

Further, the AEAD encrypted message comprises a ciphertext (e.g., encrypted ciphertext after the encryption by the second encryption algorithm), nonce (e.g., associating second decryption algo), key and associated data. The maximum length of nonce is a cipher suit specific element. The nonce and associated data are utilized in decryption of the AEAD encrypted message.

## **5. Record Protocol**

The TLS record protocol takes messages to be transmitted, fragments the data into manageable blocks, protects the records, and transmits the result. Received data is verified, decrypted, reassembled, and then delivered to higher-level clients.

TLS records are typed, which allows multiple higher-level protocols to be multiplexed over the same record layer. This document specifies four content types: handshake, application\_data, alert, and change\_cipher\_spec. The change\_cipher\_spec record is used only for compatibility purposes (see [Appendix D.4](#)).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

## **2. Protocol Overview**

Negotiating encryption algorithm

The cryptographic parameters used by the secure channel are produced by the TLS handshake protocol. This sub-protocol of TLS is used by the client and server when first communicating with each other. The handshake protocol allows peers to negotiate a protocol version, select cryptographic algorithms, optionally authenticate each other, and establish shared secret keying material. Once the handshake is complete, the peers use the established keys to protect the application-layer traffic.

<https://datatracker.ietf.org/doc/html/rfc8446>

TLS consists of two primary components:

- A handshake protocol ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering, an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

Negotiating encryption algos
- A record protocol ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

<https://datatracker.ietf.org/doc/html/rfc8446>

### 5.1. Record Layer

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of  $2^{14}$  bytes or less. Message boundaries are handled differently depending on the underlying ContentType. Any future content types MUST specify appropriate rules. Note that these rules are stricter than what was enforced in TLS 1.2.

Handshake messages MAY be coalesced into a single TLSPlaintext record or fragmented across several records, provided that:

- Handshake messages MUST NOT be interleaved with other record types. That is, if a handshake message is split over two or more records, there MUST NOT be any other records between them.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

### 5.2. Record Payload Protection

The record protection functions translate a TLSPlaintext structure into a TLSCiphertext structure. The deprotection functions reverse the process. In TLS 1.3, as opposed to previous versions of TLS, all ciphers are modeled as "Authenticated Encryption with Associated Data" (AEAD) [RFC5116]. AEAD functions provide a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and back again. Each encrypted record consists of a plaintext header followed by an encrypted body, which itself contains a type and optional padding.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

AEAD algorithms take as input a single key, a nonce, a plaintext, and "additional data" to be included in the authentication check, as described in [Section 2.1 of \[RFC5116\]](#). The key is either the client\_write\_key or the server\_write\_key, the nonce is derived from the sequence number and the client\_write\_iv or server\_write\_iv (see [Section 5.3](#)), and the additional data input is the record header.

I.e.,

```
additional_data = TLSCiphertext.opaque_type ||
                  TLSCiphertext.legacy_record_version ||
                  TLSCiphertext.length
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The AEAD approach enables applications that need cryptographic security services to more easily adopt those services. It benefits the application designer by allowing them to focus on important issues such as security services, canonicalization, and data marshaling, and relieving them of the need to design crypto mechanisms that meet their security goals. Importantly, the security of an AEAD algorithm can be analyzed independent from its use in a particular application. This property frees the user of the AEAD of the need to consider security aspects such as the relative order of authentication and encryption and the security of the particular combination of cipher and MAC, such as the potential loss of confidentiality through the MAC. The application designer that uses the AEAD interface need not select a particular AEAD algorithm during the design stage. Additionally, the interface to the AEAD is relatively simple, since it requires only a single key as input and requires only a single identifier to indicate the algorithm in use in a particular case.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.1. Authenticated Encryption

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in [Section 3.2](#), and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

## 2.2. Authenticated Decryption

Second decryption algorithm

The authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic. A ciphertext C, a nonce N, and associated data A are authentic for key K when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).

<https://datatracker.ietf.org/doc/html/rfc5116>

The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Each AEAD algorithm will specify a range of possible lengths for the per-record nonce, from N\_MIN bytes to N\_MAX bytes of input [RFC5116]. The length of the TLS per-record nonce (iv\_length) is set to the larger of 8 bytes and N\_MIN for the AEAD algorithm (see [RFC5116].. [Section 4](#)). An AEAD algorithm where N\_MAX is less than 8 bytes MUST NOT be used with TLS. The per-record nonce for the AEAD construction is formed as follows:  
<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

All handshake messages after the ServerHello are now encrypted.  
The newly introduced EncryptedExtensions message allows various extensions previously sent in the clear in the ServerHello to also enjoy confidentiality protection.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

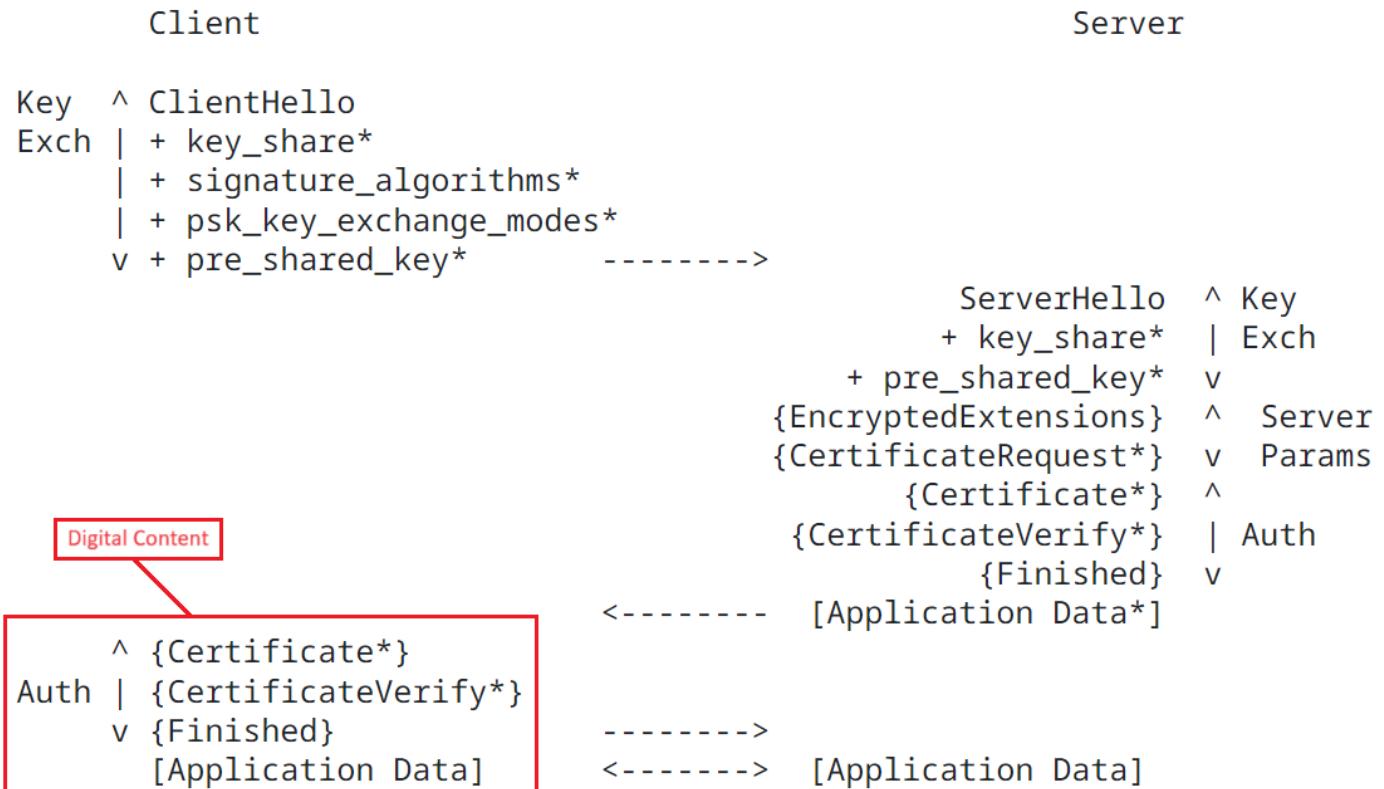
#### **4.4. Authentication Messages**

As discussed in [Section 2](#), TLS generally uses a common set of messages for authentication, key confirmation, and handshake integrity: Certificate, CertificateVerify, and Finished. (The PSK binders also perform key confirmation, in a similar fashion.) These three messages are always sent as the last messages in their handshake flight. The Certificate and CertificateVerify messages are only sent under certain circumstances, as defined below. The Finished message is always sent as part of the Authentication Block.

These messages are encrypted under keys derived from the [\[sender\]\\_handshake\\_traffic\\_secret](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

Figure 1 below shows the basic full TLS handshake:



<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2. Certificate

This message conveys the endpoint's certificate chain to the peer.

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except PSK).

The client MUST send a Certificate message if and only if the server has requested client authentication via a CertificateRequest message (Section 4.3.2). If the server requests client authentication but no suitable certificate is available, the client MUST send a Certificate message containing no certificates (i.e., with the "certificate\_list" field having length 0). A Finished message MUST be sent regardless of whether the Certificate message is empty.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.4.2.3. Client Certificate Selection

The following rules apply to certificates sent by the client:

- The certificate type MUST be X.509v3 [[RFC5280](#)], unless explicitly negotiated otherwise (e.g., [[RFC7250](#)]).
- If the "certificateAuthorities" extension in the CertificateRequest message was present, at least one of the certificates in the certificate chain SHOULD be issued by one of the listed CAs.
- The certificates MUST be signed using an acceptable signature algorithm, as described in [Section 4.3.2](#). Note that this relaxes the constraints on certificate-signing algorithms found in prior versions of TLS.
- If the CertificateRequest message contained a non-empty "oid\_filters" extension, the end-entity certificate MUST match the extension OIDs that are recognized by the client, as described in [Section 4.2.5](#).

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### [4.4.3. Certificate Verify](#)

This message is used to provide explicit proof that an endpoint possesses the private key corresponding to its certificate. The CertificateVerify message also provides integrity for the handshake up to this point. Servers MUST send this message when authenticating via a certificate. Clients MUST send this message whenever authenticating via a certificate (i.e., when the Certificate message is non-empty). When sent, this message MUST appear immediately after the Certificate message and immediately prior to the Finished message.

Structure of this message:

```
struct {
    SignatureScheme algorithm;
    opaque signature<0..2^16-1>;
} CertificateVerify;
```

The algorithm field specifies the signature algorithm used (see [Section 4.2.3](#) for the definition of this type). The signature is a digital signature using that algorithm. The content that is covered under the signature is the hash output as described in [Section 4.4.1](#), namely:

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

#### 4.3.2. Certificate Request

A server which is authenticating with a certificate MAY optionally request a certificate from the client. This message, if sent, MUST follow EncryptedExtensions.

Structure of this message:

```
struct {
    opaque certificate_request_context<0..2^8-1>;
    Extension extensions<2..2^16-1>;
} CertificateRequest;
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

`certificate_request_context`: An opaque string which identifies the certificate request and which will be echoed in the client's Certificate message. The `certificate_request_context` MUST be unique within the scope of this connection (thus preventing replay of client CertificateVerify messages). This field SHALL be zero length unless used for the post-handshake authentication exchanges described in [Section 4.6.2](#). When requesting post-handshake authentication, the server SHOULD make the context unpredictable to the client (e.g., by randomly generating it) in order to prevent an attacker who has temporary access to the client's private key from pre-computing valid CertificateVerify messages.

`extensions`: A set of extensions describing the parameters of the certificate being requested. The "signature\_algorithms" extension MUST be specified, and other extensions may optionally be included if defined for this message. Clients MUST ignore unrecognized extensions.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4.3.2>

- RSASSA-PSS signature schemes are defined in [Section 4.2.3](#).
- The "supported\_versions" ClientHello extension can be used to negotiate the version of TLS to use, in preference to the legacy\_version field of the ClientHello.
- The "signature\_algorithms\_cert" extension allows a client to indicate which signature algorithms it can validate in X.509 certificates.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

If sent by a client, the signature algorithm used in the signature MUST be one of those present in the supported\_signature\_algorithms field of the "signature\_algorithms" extension in the CertificateRequest message.

In addition, the signature algorithm MUST be compatible with the key in the sender's end-entity certificate. RSA signatures MUST use an RSASSA-PSS algorithm, regardless of whether RSASSA-PKCS1-v1\_5 algorithms appear in "signature\_algorithms". The SHA-1 algorithm MUST NOT be used in any signatures of CertificateVerify messages.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

#### 4.2.3. Signature Algorithms

TLS 1.3 provides two extensions for indicating which signature algorithms may be used in digital signatures. The "signature\_algorithms\_cert" extension applies to signatures in certificates, and the "signature\_algorithms" extension, which originally appeared in TLS 1.2, applies to signatures in CertificateVerify messages. The keys found in certificates MUST also be of appropriate type for the signature algorithms they are used with. This is a particular issue for RSA keys and PSS signatures, as described below. If no "signature\_algorithms\_cert" extension is present, then the "signature\_algorithms" extension also applies to signatures appearing in certificates. Clients which desire the server to authenticate itself via a certificate MUST send the "signature\_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature\_algorithms" extension, then the server MUST abort the handshake with a "missing\_extension" alert (see [Section 9.2](#)).

The "signature\_algorithms\_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. Implementations which have the same policy in both cases MAY omit the "signature\_algorithms\_cert" extension.

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

As shown below, the receiving party will be able to decrypt the encrypted message with the provided signature decryption algorithm information i.e., SHA-256 RSA decryption algorithm.

We are now prepared to show that we can decrypt encrypted messages. We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

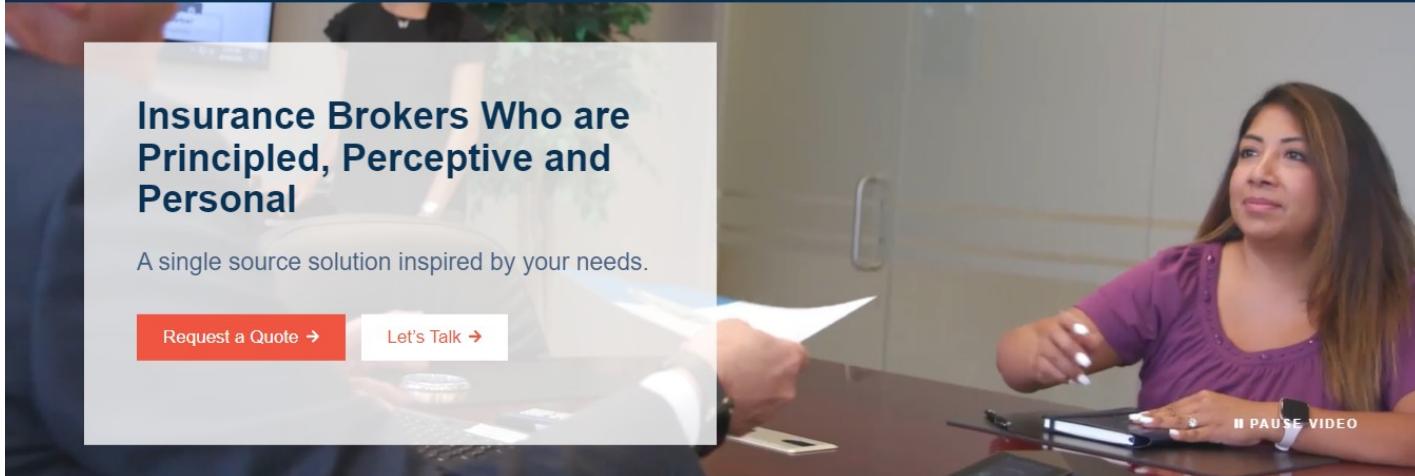
First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

|  |  |
|--|--|
|  | <p>The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A <i>cryptographic hash function</i> is a function that computes a <i>message authentication code</i> from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that <math>H</math> is a cryptographic hash function. To sign a message <math>m</math>, party A computes <math>h = D_A(H(m))</math> and sends <math>E_B(m, h)</math> to B. Party B now has evidence that A signed <math>m</math> because <math>E_A(h) = H(m)</math>, and A is the only one who could have generated a value <math>h</math> with that property.</p> |
| 39. The software system or computer program of claim 38, wherein the decrypting is done using a key associated with each decryption algorithm. | The standard practices the method such that the decrypting is done using a key (e.g., decryption key) associated with each decryption algorithm (e.g., signature decryption algorithm such as SHA-256RSA, etc., and AEAD decryption algorithm such as TLS_AES_256_GCM_SHA384, etc.).   |

The Higginbotham Difference About Us Careers Blog Locations Become a Higginbotham Partner  Report a Claim

 **Higginbotham** \* Business Insurance Employee Benefits HR Services Financial Services Personal Insurance Captive Insurance Let's Talk



Insurance Brokers Who are Principled, Perceptive and Personal

A single source solution inspired by your needs.

[Request a Quote →](#) [Let's Talk →](#)

<https://www.higginbotham.com/>

II PAUSE VIDEO

The screenshot shows the Google Play Store search results for the query "higginbotham". The top result is the "Higginbotham FSA" app, which is described as "Higginbotham Wex Health Mobile". It features a red box around its listing. Below the app are three promotional screenshots of the mobile application interface, showing account details, activity history, and expense management.

<https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US>

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP-Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 18B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB 89 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
crls_size(0x3ada) 00
```

*Source: Fiddler Capture*

As shown below, the signature decryption algorithm utilizes a private key for a first decryption and the AEAD decryption algorithm uses a key K. Both the decryption techniques are decrypting using their respective associated keys.

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|  |  |
|--|--|
|  | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 40. The software system or computer program of claim 39, wherein the key is resident in hardware of the target unit or the key is retrieved from a | The standard utilized by the accused instrumentality practices the method such that the key is resident in hardware (e.g., stored in a memory storage of the server such as a database, RAM, etc.) of the target unit (e.g., server of the accused instrumentality) or the key is retrieved from a server.   |

server.

The screenshot shows the Higginbotham website homepage. At the top, there is a navigation bar with links to "The Higginbotham Difference", "About Us", "Careers", "Blog", "Locations", "Become a Higginbotham Partner", a search icon, and a "Report a Claim" button. Below the navigation bar is a main menu with categories: "Business Insurance", "Employee Benefits", "HR Services", "Financial Services", "Personal Insurance", "Captive Insurance", and a "Let's Talk" button. A video player is prominently displayed in the center, showing a woman in a purple shirt sitting at a desk, looking towards the camera. The video has a caption that reads "Insurance Brokers Who are Principled, Perceptive and Personal" and a subtitle "A single source solution inspired by your needs." Below the video are two buttons: "Request a Quote" and "Let's Talk". The URL <https://www.higginbotham.com/> is visible at the bottom of the page.

The Google Play Store search results for "higginbotham" are displayed. The top result is the "Higginbotham FSA" app, which is part of the "Higginbotham Wex Health Mobile" suite. The app has a red border around its listing. It shows 1K+ downloads and is rated E for Everyone. A large "Install" button is present. To the right of the app listing are three screenshots of the app's interface: one showing account details, one showing account activity, and one showing expense management.

<https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US>

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
    extended_master_secret empty
    ec_point_formats uncompressed [0x0]
    status_request OCSP-Implicit Responder
    psk_key_exchange_modes 01 01
    renegotiation_info 00
    0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 58 6E
2D 8A 88 BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 FB 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
    server_name www.higginbotham.com
    preSharedKey 00
```

*Source: Fiddler Capture*



## **Server hardware guide: Architecture, products and management**

### **3. Random access memory**



RAM is the main type of memory in a computing system.

RAM holds the software instructions and data needed by the processor, along with any output from the processor, such as data to be moved to a storage device. Thus, RAM works very closely with the processor and must match the processor's incredible speed and performance. This kind of fast memory is usually termed dynamic RAM, and several DRAM variations are available for servers.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>



## Server hardware guide: Architecture, products and management

### 4. Hard disk drive



This hardware is responsible for reading, writing and positioning of the hard disk, which is one technology for data storage on server hardware. Developed at IBM in 1953, the [hard disk drive \(HDD\)](#) has evolved over time from the size of a refrigerator to the standard 2.5-inch and 3.5-inch form factors.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>

As shown below, the server comprises a memory storage to store messages for establishing secure TLS communication. the standard discloses multiple signature encryption algorithms for a first encryption and multiple AEAD encryption algorithms for the second encryption. A signature decryption algorithm utilizes a private key for decrypting the first bitstream encrypted with the signature encryption and an AEAD decryption algorithm uses a key K for decrypting the second bitstream encrypted with the AEAD encryption. Both the decryption techniques are decrypting using their respective associated keys. A server must have a storage to store information pertaining to these algorithms and their corresponding keys such as private key, Key K, etc., to establish secure TLS communication with a client.

Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. This is necessary for the ClientHello storage mechanism described in Section 8.2 because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.

<https://datatracker.ietf.org/doc/html/rfc8446#>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

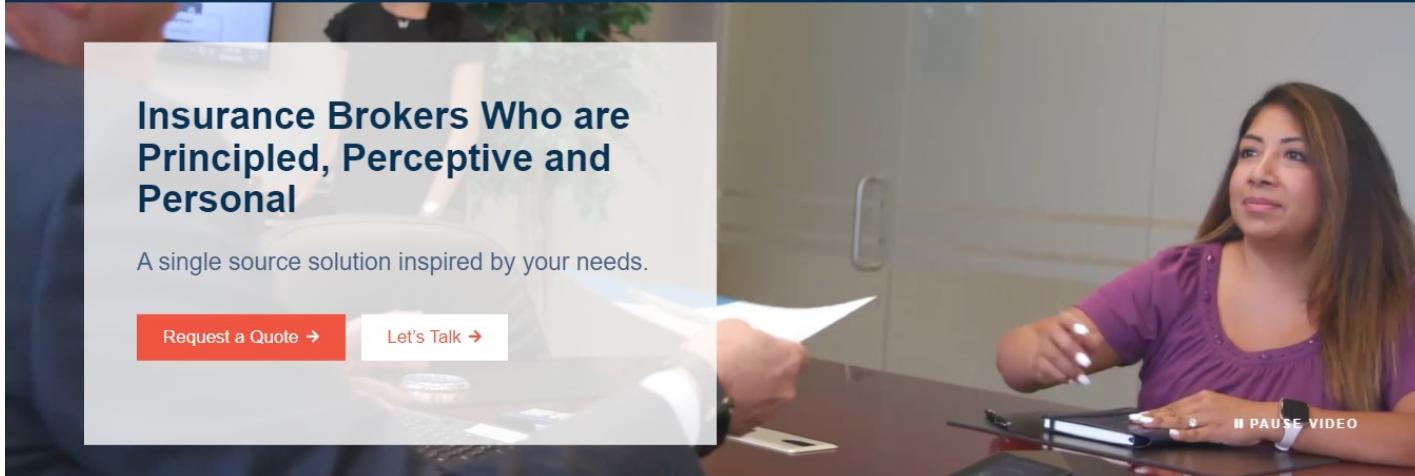
| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|  |  |
|--|--|
|  | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 41. The software system or computer program of claim 40, wherein the key is contained in a key data structure. | The standard utilized by the accused instrumentality practices the method such that the key (e.g., private key, Key K, etc.) is contained in a key data structure (e.g., data structure).  |

The Higginbotham Difference About Us Careers Blog Locations Become a Higginbotham Partner  Report a Claim

 Higginbotham\* Business Insurance Employee Benefits HR Services Financial Services Personal Insurance Captive Insurance Let's Talk



Insurance Brokers Who are Principled, Perceptive and Personal

A single source solution inspired by your needs.

Request a Quote → Let's Talk →

<https://www.higginbotham.com/>

II PAUSE VIDEO

The image shows a screenshot of the Google Play Store search results for "higginbotham". The search bar at the top contains the query. Below it, there are three categories: "Apps & games", "Movies & TV", and "Books". The first result is the "Higginbotham FSA" app, which is part of the "Higginbotham Wex Health Mobile" package. It has a red border around its listing. The app has a download count of over 1K+ and is rated "Everyone". A large "Install" button is visible. To the right of the app listing are three screenshots of the app's interface: one showing accounts, one showing account activity, and one showing a dashboard with expense management options.

<https://play.google.com/store/search?q=higginbotham&c=apps&hl=en&gl=US>

```
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP-Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 86 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 88 BD F9 05 0F D4 63 E1 FC 35 82 4F 90 98 5C AB 89 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 BF 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 3C D6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
crls_size(0x3ada) 00
```

*Source: Fiddler Capture*

The accused instrumentality utilizes a server to establish a secure TLS communication with a client. The server must comprise a memory storage and store data according to a data structure to implement the standard efficiently.



## **Server hardware guide: Architecture, products and management**

### **3. Random access memory**



RAM is the main type of memory in a computing system.

RAM holds the software instructions and data needed by the processor, along with any output from the processor, such as data to be moved to a storage device. Thus, RAM works very closely with the processor and must match the processor's incredible speed and performance. This kind of fast memory is usually termed dynamic RAM, and several DRAM variations are available for servers.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>



## Server hardware guide: Architecture, products and management

### 4. Hard disk drive



This hardware is responsible for reading, writing and positioning of the hard disk, which is one technology for data storage on server hardware. Developed at IBM in 1953, the [hard disk drive \(HDD\)](#) has evolved over time from the size of a refrigerator to the standard 2.5-inch and 3.5-inch form factors.

<https://www.techtarget.com/searchdatacenter/feature/Drill-down-to-basics-with-these-server-hardware-terms>

A data structure is a specialized format for organizing, processing, retrieving and storing data.

There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose. Data structures make it easy for users to access and work with the data they need in appropriate ways. Most importantly, data structures frame the organization of information so that machines and humans can better understand it.

In computer science and computer programming, a data structure may be selected or designed to store data for the purpose of using it with various algorithms. In some cases, the algorithm's basic operations are tightly coupled to the data structure's design. Each data structure contains information about the data values, relationships between the data and -- in some cases -- functions that can be applied to the data.

<https://www.techtarget.com/searchdatamanagement/definition/data-structure>

As shown below, the server comprises a memory storage to store messages for establishing secure TLS communication. the standard discloses multiple signature encryption algorithms for a first encryption and multiple AEAD encryption algorithms for the second encryption. A signature decryption algorithm utilizes a private key for decrypting the first bitstream encrypted with the signature encryption and an AEAD decryption algorithm uses a key K for decrypting the second bitstream encrypted with the AEAD encryption. Both the decryption techniques are decrypting using their respective associated keys. A server must have a storage to store information pertaining to these algorithms and their corresponding keys such as private key, Key K, etc., to establish secure TLS communication with a client.

Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. This is necessary for the ClientHello storage mechanism described in [Section 8.2](#) because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.

<https://datatracker.ietf.org/doc/html/rfc8446#>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

We are now prepared to show that we can decrypt encrypted messages.

We can find a pair of large prime numbers  $p$  and  $q$ , compute  $n = pq$  and  $\varphi(n) = (p - 1)(q - 1)$ , find  $d$  which is relatively prime to  $\varphi(n)$ , and compute the value  $e$  for which  $de \equiv 1 \pmod{\varphi(n)}$ . We know that  $de - 1$  is divisible by  $\varphi(n)$ , so there is a number  $k$  satisfying  $de - 1 = k\varphi(n)$ .

Recall from Section II that  $(e, n)$  is the encryption key and  $(d, n)$  is the decryption key. If  $m$  is a plaintext message, then the ciphertext is

$$c = m^e \pmod{n}.$$

First encryption

To decrypt, we compute  $c^d \pmod{n}$  to obtain

$$c^d \pmod{n} = (m^e \pmod{n})^d \pmod{n} = m^{de} \pmod{n} = m^{1+k\varphi(n)} \pmod{n}.$$

The result of Exercise 3.13 tells us that

$$m \equiv m^{1+k\varphi(n)} \pmod{n},$$

First decryption

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

|  |  |
|--|--|
|  | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 47. The software system or computer program of claim 39, wherein each encryption algorithm is a symmetric key system or an | The standard practices the method such that each encryption algorithm (e.g., signature encryption algorithm i.e., SHA256RSA, etc., and AEAD encryption algorithm i.e., TLS_AES_256_GCM_SHA384, etc.) is a symmetric key system (e.g., AEAD encryption algorithm, etc.) or an asymmetric key system (e.g., signature encryption algorithm). <p>As shown below, the server comprises a memory storage to store messages for</p>  |

|                    |  |
|--------------------|--|
| asymmetric system. | key establishing secure TLS communication. the standard discloses multiple signature encryption algorithms for a first encryption and multiple AEAD encryption algorithms for the second encryption. A signature decryption algorithm utilizes a private key for decrypting the first bitstream encrypted with the signature encryption and an AEAD decryption algorithm uses a key K for decrypting the second bitstream encrypted with the AEAD encryption. The standard defines the signature encryption algorithm as an asymmetric cryptography algorithm and the AEAD encryption algorithm as the symmetric cryptography algorithm.<br><br>Because the ClientHello indicates the time at which the client sent it, it is possible to efficiently determine whether a ClientHello was likely sent reasonably recently and only accept 0-RTT for such a ClientHello, otherwise falling back to a 1-RTT handshake. <u>This is necessary for the ClientHello storage mechanism described in Section 8.2</u> because otherwise the server needs to store an unlimited number of ClientHellos, and is a useful optimization for self-contained single-use tickets because it allows efficient rejection of ClientHellos which cannot be used for 0-RTT.<br><a href="https://datatracker.ietf.org/doc/html/rfc8446#">https://datatracker.ietf.org/doc/html/rfc8446#</a><br><br>Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated. Authentication can happen via <u>asymmetric cryptography</u> (e.g., RSA [RSA], the Elliptic Curve Digital Signature Algorithm (ECDSA) [ECDSA], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032]) or a symmetric pre-shared key (PSK).<br><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-4">https://datatracker.ietf.org/doc/html/rfc8446#section-4</a> |
|--------------------|--|

cipher\_suites: A list of the symmetric cipher options supported by the client, specifically the record protection algorithm (including secret key length) and a hash to be used with HKDF, in descending order of client preference. Values are defined in [Appendix B.4](#). If the list contains cipher suites that the server does not recognize, support, or wish to use, the server MUST ignore those cipher suites and process the remaining ones as usual. If the client is attempting a PSK key establishment, it SHOULD advertise at least one cipher suite indicating a Hash associated with the PSK.

<https://datatracker.ietf.org/doc/html/rfc8446#section-4>

The "extension\_data" field of these extensions contains a SignatureSchemeList value:

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),
```

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

There is also a decryption function  $D$  that takes a ciphertext and a decryption key  $K_D$ , and reproduces the plaintext message.

$$D(C, K_D) = P$$

In a *symmetric* or *private key* system, the encryption and decryption keys are the same. A private key system has the disadvantage that the parties must get together and agree upon a shared key. It has the advantage in that the computational overhead is smaller. Once the key is in place, communication can happen much faster.

In an *asymmetric* or *public key* system, the two keys are different. Each participant has her or his own pair of keys. The encryption keys are known to everyone, but the decryption keys are kept secret. Person  $A$  can look up person  $B$ 's encryption key, encrypt a message with it, and send the result to person  $B$ . Only someone with  $B$ 's decryption key, namely only  $B$ , can read the message. An eavesdropper  $E$  might intercept the encrypted message but would not be able to decipher it.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party A computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to B. Party B now has evidence that A signed  $m$  because  $E_A(h) = H(m)$ , and A is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

This specification defines the following cipher suites for use with TLS 1.3.

| Description                  | Value       |
|------------------------------|-------------|
| TLS_AES_128_GCM_SHA256       | {0x13,0x01} |
| TLS_AES_256_GCM_SHA384       | {0x13,0x02} |
| TLS_CHACHA20_POLY1305_SHA256 | {0x13,0x03} |
| TLS_AES_128_CCM_SHA256       | {0x13,0x04} |
| TLS_AES_128_CCM_8_SHA256     | {0x13,0x05} |

<https://datatracker.ietf.org/doc/html/rfc8446#section-1>

The authenticated encryption operation has four inputs, each of which is an octet string:

A secret key K, which MUST be generated in a way that is uniformly random or pseudorandom.

A nonce N. Each nonce provided to distinct invocations of the Authenticated Encryption operation MUST be distinct, for any particular value of the key, unless each and every nonce is zero-length. Applications that can generate distinct nonces SHOULD use the nonce formation method defined in Section 3.2, and MAY use any other method that meets the uniqueness requirement. Other applications SHOULD use zero-length nonces.

A plaintext P, which contains the data to be encrypted and authenticated.

The associated data A, which contains the data to be authenticated, but not encrypted.

<https://datatracker.ietf.org/doc/html/rfc5116>

|   |  |
|---|--|
|   | <p><b><u>2.2. Authenticated Decryption</u></b></p> <p>The <u>authenticated decryption operation has four inputs: K, N, A, and C, as defined above. It has only a single output, either a plaintext value P or a special symbol FAIL that indicates that the inputs are not authentic.</u> A ciphertext C, a nonce N, and associated data A are <u>authentic for key K</u> when C is generated by the encrypt operation with inputs K, N, P, and A, for some values of N, P, and A. The authenticated decrypt operation will, with high probability, return FAIL whenever the inputs N, P, and A were crafted by a nonce-respecting adversary that does not know the secret key (assuming that the AEAD algorithm is secure).</p> <p><a href="https://datatracker.ietf.org/doc/html/rfc5116">https://datatracker.ietf.org/doc/html/rfc5116</a></p> <p>The AEAD output consists of the ciphertext output from the AEAD encryption operation. The length of the plaintext is greater than the corresponding TLSPlaintext.length due to the inclusion of TLSInnerPlaintext.type and any padding supplied by the sender. The <u>length of the AEAD output will generally be larger than the plaintext, but by an amount that varies with the AEAD algorithm.</u></p> <p><a href="https://datatracker.ietf.org/doc/html/rfc8446#section-1">https://datatracker.ietf.org/doc/html/rfc8446#section-1</a></p> |
| 48. The software system or computer program of claim 39, further translatable for associating a first Message Authentication Code | The standard practices associating a first Message Authentication Code (MAC) (e.g., message authentication code with hashing function) or first digital signature with each encrypted bit stream (e.g., encrypted bit stream with the signature encryption algorithm i.e., SHA256RSA, etc., and encrypted bitstream with the AEAD encryption algorithm i.e., TLS_AES_256_GCM_SHA384, etc.).  |

(MAC) or first digital signature with each encrypted bit stream.

algorithm. It performs a message authentication code with the utilized hashing function.

```

SignedCertTimestamp (RFC6962) empty
ALPN h2, http/1.1
signature_algs ecdsa_secp256r1_sha256, rsa_pss_rsae_sha256, rsa_pkcs1_sha256, ecdsa_secp384r1_sha384, rsa_pss_rsae_sha384, rsa_pkcs1_sha384, rsa_pss_rsae_sha512, rsa_pkcs1_sha512
0x001b 02 00 02
supported_groups grease [0x5a5a], unknown [0x6399], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]
key_share 04 ED 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 32 5D 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 8E 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85
7A 95 44 34 07 19 C8 33 3F EC 9C 45 46 0B 84 C9 1C 22 E1 3C 9F 87 71 A3 5B C3 72 80 E1 C9 5D CA 9D C4 69 B0 05 5B 9D E2 4B 2B 56 F5 BF 82 84 67 9C C6 65 5F 82 3E B2 F0 BB F5 3B AF 0F 86 11 A4 1C 00 5A 46 68 D9 57 AF 31 C6 C2 F4 F0 38 F1
EC 97 6A 21 78 C4 A8 AA B4 85 CE C9 85 67 D7 7A 0C EB AB 79 1D 83 3A B2 3B 60 D6 A3 C2 01 87 A7 F7 CC B8 A0 E5 1E 1F 35 88 34 65 2F BA CA 06 9C 3B 17 9D A6 11 B2 39 9F 34 00 6C DB D0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E 2F
C0 61 1D A0 54 55 73 A1 99 23 8C 39 BA A0 C6 73 7A 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A B5 2C 21 80 CA 1F E2 AC B4 73 A5 FD EB 7B 6D B5 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F E0 75 BA 39 AA 0B 66 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F
23 C1 C3 B1 5E E8 05 13 61 7C 42 DC 1C 3F F0 AB 47 A3 00 28 E6 A7 C1 C1 26 7B D7 FB 9F E9 09 14 5A E3 B6 5A 5B 9B D2 D5 47 0D 6B 7B 4A 71 6D 7E 01 37 F8 E0 A3 1E 58 6B D6 D2 C0 26 53 B8 9B B0 09 DF 5A 20 22 E9 A3 DD 00 9E 98 F0 4C A0
11 99 AD C0 48 DA D1 37 3B 2A AF 72 43 91 BD 66 6E 23 56 C7 5A E0 5E 74 33 F7 3A 86 40 07 F3 7D 95 B1 38 70 23 05 30 1B 5E A5 C7 07 13 84 84 A9 90 B3 66 B5 11 A9 A6 72 3C 15 DC EC 94 3E 55 93 83 64 C0 BF 79 2F C3 22 6C B1 C4 AB 5B 12
83 A0 B0 8F 26 98 2F 47 47 7B 2B 10 49 21 80 5F 14 53 0B 4E DA 90 E3 A0 82 98 C8 89 6E 63 TA 6A AA A7 E6 C8 08 2D 4B 10 BA 70 A0 3A 93 D1 75 07 59 1C 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 99 3C E8 A3 A7 F0 73 D4 FA 81 9F 99 BF 70 B7 A1
08 BB 58 A8 C2 2D C1 FB BB B7 23 89 CB F0 92 B8 A6 85 B3 E4 6C 6B 59 20 88 B5 53 12 EB 49 EB A3 13 B3 78 A5 F1 3A 70 86 8B 8B 96 96 7F 23 9C 37 19 0C C0 9C 0B 14 8F 67 78 4A F5 03 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 A4 AB
CA 6B 3C A4 85 1D 6E 70 74 88 C7 9D B9 8B 4D DD 4B 39 6D AB 00 9C 52 17 C3 C4 AD F6 10 A0 DE 0A B5 86 C7 49 SD 94 3A C5 39 95 02 BC 1E A2 73 6A 2D 16 1C 54 31 68 1D A6 C9 B5 C3 A0 12 3B 1C 72 10 60 3E 3B 23 6A 68 92 DF CC 86 91 E7 9E
2B 72 01 FC C4 34 07 61 05 82 53 94 E6 2C CC 30 74 A3 A7 08 41 97 77 A4 14 5C 4D 4F E2 68 66 94 AD 20 34 22 4F 29 2B 29 37 42 25 08 53 92 F4 CD A9 19 3A 44 8A 23 98 EC 22 E9 01 24 6D 29 8B DC 08 B0 45 B9 45 88 43 BA E4 99 3A 32 23 32 6C
31 26 FC 06 97 C2 2E 7D 39 B9 AA 17 69 3C 94 66 FA C3 3B C9 E4 5C 75 BC 59 8B 60 14 A7 26 83 FF 72 2C 0D F7 4F E9 88 B0 33 67 6D F3 C2 C0 1A 5B 69 2F 61 48 53 94 9A F6 67 00 03 7A 8C 8E BC 4E 4C 2C 49 E0 82 8C DA 88 AF DA 6A 6D B5 08
9D CA A9 A5 5A E2 91 95 3C 70 5A 3C B0 B2 AD CE 61 A9 8F C4 54 15 8B 3C A9 31 21 D8 28 CD AC 99 09 A2 37 B4 46 53 2F AF E4 AF E8 F2 55 CC FC 4F 48 78 A4 FC 28 64 10 17 C6 61 71 2F 90 25 26 E0 C5 38 2A C6 4C A4 99 9E F8 D6 3B 1C
48 C0 AE DC B0 B6 1C 85 B3 75 4F B4 5A 14 3F C0 90 CD E6 B7 18 12 6E 34 52 C1 B5 A3 1B E4 17 72 1F F8 25 20 88 21 6B 1A C3 E4 A2 53 6C CC 0A 82 32 07 76 C5 23 8C C0 40 07 00 C0 B0 C2 4A 48 D3 B7 13 00 11 C5 88 73 26 F2 EA 96 15 55
CC 3E EB 0F 80 28 93 D4 4B 1D 52 18 44 B8 86 BD F2 BC 1E 43 34 23 C8 57 4B 65 2F E6 29 97 C8 39 39 BD E7 0F 2D C2 AF 53 DA AA 56 C1 7D 4B 93 54 D9 67 C5 D9 F5 1E 7B 86 49 57 45 B5 39 60 62 40 C2 96
B1 24 9A DA B4 8D FD 8A 9C SC C0 87 1E 19 12 FD FA 32 DB 69 C3 B3 T8 76 84 15 70 C9 69 73 7A F6 A7 D1 21 4C F0 63 06 42 37 TD F3 3A 85 EF D6 02 FD F1 6D 4A 3C 55 AB BA 8A 0F 45 A6 5C C8 53 19 F6 A1 75 06 90 A0 26 2D A3 90
9B AB 18 53 2C E9 0F 05 83 3D 0E 49 76 33 D8 4D 8E 72 29 2F FB 51 3B 42 5F 76 C4 AE 54 45 E6 91 8D 2B 45 F0 9A 1E 65 C0 3A D7 99 F5 AE F1 9D E9 45 D3 A3 C3 DB 7A 74 E0 3D ED 7F 6D 00 1D 00 20 FA D2 73 2A 7D BB 08 13 72 9B 38 A0 89 F4
7A 36 23 01 54 D1 F6 FB F9 09 92 32 D7 CC C6 34 38 24
extended_master_secret empty
ec_point_formats uncompressed [0x0]
status_request OCSP - Implicit Responder
psk_key_exchange_modes 01 01
renegotiation_info 00
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 0F 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 6B B6 62 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 CD 58 6E
2D 8A 8B BD F9 05 0F D4 63 E1 FC 35 82 4F 90 88 3C AB B9 04 52 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 FB 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C
E4 C8 4A D5 F3 C6 8F 9A DF C3 0A 9F 27 E4 C6 05 53 5F 83 38 B9 5A
server_name www.higginbotham.com
osrae (0x3a3a) 00

```

*Source: Fiddler Capture*

SignedCertTimestamp (RFC6962) empty  
ALPN h2, http/1.1  
signature algs ecdsa secp256r1 sha256, rsa\_pss\_rsae\_sha256, rsa\_pkcs1\_sha256, ecdsa secp384r1 sha384, rsa\_pss\_rsae\_sha384, rsa\_pkcs1\_sha384, rsa\_pss\_rsae\_sha512, rsa\_pkcs1\_sha512  
0x001b 02 00 02  
supported\_groups grease [0x5a5a], unknown [0x6399], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]  
key\_share 04 ED 5A 5A 00 01 00 63 99 04 C0 73 FD 09 6A 32 61 2F 32 5D 09 69 67 98 9F 8B 88 D9 A0 C6 15 CC 37 BB 51 91 1D 30 CE 8D F6 E8 5F 54 C2 5A E3 89 2C 7B 89 AA BB C5 AF E9 56 10 B8 15 57 94 A1 A1 0C 2A 73 3E B4 85  
7A 95 44 34 07 19 C8 33 3F BC 93 45 08 B4 94 C1 22 E1 93 49 8F 77 13 A5 B3 72 B0 C1 C9 D5 CA 69 80 05 9B 92 E4 2A 28 56 F9 6C 65 5F 82 E2 B0 F5 B3 AB 0F 86 11 A4 1C 00 54 46 89 D9 57 AF 31 C6 C2 F4 F0 38 F1  
EC 97 6A 21 28 C4 A8 AA B4 85 C9 C8 95 67 D7 7A 0C EB AB 79 1D 83 3A B2 3B 60 D6 A3 C2 01 87 A7 F7 CC 88 A0 E5 1E 0F 55 88 34 65 2F BA CA 06 9C 38 17 9D A6 11 B2 39 9F 34 00 6C CB D0 53 3E 57 16 06 E2 72 B7 D8 42 36 EA 03 9B DB 8E 2F  
C0 61 1D A0 54 55 73 A1 99 23 8C 39 BA A0 C6 73 TA 95 49 8A 6B 1A 69 74 99 49 55 5B BC 0A B5 2C 21 80 CA 1F E2 AC 84 73 A5 FD EB 7B 6D B3 93 D6 10 37 A1 63 C8 18 C6 49 D3 71 0F E0 75 BA 39 AA 0B 86 56 BD DC 3C 1E 5D 8A 7E 1A B6 2E F5  
23 C1 03 0B 5E 08 13 61 C2 4D 1C 3F B0 47 A3 00 28 E6 A7 C1 C1 26 7B D7 FB 9F E9 09 14 5A E3 B6 5A 5B 9B D2 D5 47 0D 6B 7B 4A 71 6D 7E 01 37 F0 E0 A3 1E 58 6B 06 D2 C0 26 53 B3 98 B0 09 DF 5A 20 22 E9 A3 DD 00 E6 98 F0 4C A0  
11 99 AD C0 48 DA D1 37 3B 2A AF T2 43 91 BD 66 8E 23 56 C7 5A E0 5E 74 33 BF 3A 86 40 07 F3 7D 95 B1 38 70 23 05 30 1E 5E A5 C7 07 13 84 84 A9 80 B3 66 B5 11 A9 A8 61 72 3C 15 DC EC 94 3E 55 93 83 64 CO BF 79 2F C3 22 6C B1 C4 AB 5B 12  
83 A0 B0 8F 26 98 2F 47 7B 20 81 5F 14 53 04 BE A0 E3 82 98 C8 89 6E 73 AA 7A A8 1E 7C 08 2D 4B 10 BA 70 A0 3A 93 01 75 07 99 7D C8 90 55 26 AB 84 90 91 C9 72 10 EB 82 98 C8 E3 A7 F0 73 DA 4F 81 9F 99 BF 77 B7 A1  
08 BB 58 A8 CC 2D C1 FB BR B7 23 89 CF 90 B2 AB 65 B3 4F 6C 6B 59 20 88 B5 12 49 FB 4A 13 B3 7A 85 F1 3A 70 86 8B 96 7F 23 9C 37 19 0C C0 3C 08 14 4F 67 78 4A F5 03 04 61 CB E4 AD 37 D0 65 57 26 8B 24 DC 8E 15 A4 AB  
CA 6B 3C 4A 85 1D 6E 70 74 86 C7 9D B9 8B 4D DD 4B 39 6D AB 00 9C 52 17 C3 C3 AD F6 10 A0 DE 0A B5 86 C7 49 5D 94 3A C5 39 95 02 BC 1E A2 73 6A 2D 16 1C 54 31 68 1D A6 C9 B5 C3 A0 12 3B 1C 72 10 60 3E 2B 3D 6A 68 92 DF CC 86 91 E7 98  
2B 72 01 TB C4 34 07 61 59 34 93 H4 36 2C CC 30 74 A3 7A 08 47 91 77 14 5C 28 6B 66 94 20 34 22 4F 29 2B 29 37 23 08 53 92 F4 CD A5 19 3A 44 8A 23 98 EC 22 E9 01 24 6D 29 BB 8C 45 88 43 8A E4 99 3A 32 23 6C 66  
31 26 FC 06 97 C2 2E TD 39 B9 AA 17 69 3C 94 66 FA C3 CB 9E 45 7C 5B 59 8B 60 14 A7 26 83 FF 72 2C 0D F4 E9 88 03 33 67 FD F3 C2 C0 1A 5B 69 2F 61 48 53 94 RA F4 67 00 03 7A 8C BE 4B 4C 2C 49 E0 82 8C DA 88 AF DA 6D B5 08  
9D CA 09 5A M2 91 50 7A 5C 3C D2 0D CE 61 B4 54 15 8B 3C A9 32 21 D8 2C AC 99 09 A2 37 B4 46 53 2F AE 4F E8 F2 55 CC FC 4F 4B 78 A4 FC 28 64 10 17 C6 61 71 2F 90 25 26 E0 C5 38 2A C6 4C A4 99 9E F8 D8 3B 1C  
48 CO AE DC B0 86 1C 85 83 75 4F BA 54 14 3F C0 90 CD E6 87 18 12 6E 34 52 C1 B5 A3 1B E4 17 72 1F F8 25 20 88 21 6B 1A 3C E4 4A 2E 53 6C CO DA 82 32 07 W6 C5 23 8C CO 40 07 00 CO B0 C2 4A 48 D3 B7 13 00 11 C5 88 73 26 F2 EA 96 15 55  
C0 3E EB OF 08 28 93 D4 49 B4 1D 52 18 44 88 88 6D F5 BC AD 88 60 BF F2 BC 1E 43 34 23 C8 57 48 65 2F E6 29 97 C8 39 3B ED 70 F2 D2 C2 AF 53 DA AA 56 C1 7D 48 93 54 D9 67 C5 D9 F5 1E 7B 86 49 57 45 85 58 56 84 32 42 85 39 60 62 40 C2 96  
B1 24 9A DA B4 8D FD 8A 5C C0 87 19 12 FD FA 32 FD 69 C3 B3 7B 76 84 15 70 C9 69 73 7A F6 A6 D7 A1 21 4C F0 63 06 42 37 FD F3 A3 B5 D0 85 EF D6 02 FD F1 6D 4A 3C 55 AB BA 8A 0F 45 A6 5C C8 53 19 F6 A1 75 06 90 A0 26 2D A3 90  
9B AB 18 53 2C E9 OF 05 83 3D DE 49 76 33 D8 4D 8E 72 29 2F FB 51 3B 42 5F 76 C4 AE 54 45 E6 91 8D 2B 45 F0 9A 1E 65 C0 3A D7 99 F5 AE F1 9D E9 45 D3 A3 CD DB 7A 74 03 D7 ED F7 6D 00 01 20 FA D2 73 2A 7D BB 08 13 72 9B 38 A0 89 F4  
7A 38 23 01 54 D1 F6 FB F9 92 32 D7 C6 C4 34 24  
  
extended\_master\_secret empty  
ec\_point\_formats uncompressed [0x0]  
status\_request OCSP - Implicit Responder  
psk\_key\_exchange\_modes 01 01  
renegotiation\_info 00  
0xfe0d 00 00 01 00 01 8B 00 20 08 8F C0 F2 2E C4 97 D9 A5 41 74 90 DC B9 00 DF 96 AC CE 3B D4 B6 C4 68 48 15 D0 44 C5 08 4A 5E 00 D0 A7 B0 A8 9E C4 E4 1F BB 61 CD 40 13 BB 50 B2 C5 69 CC 12 85 91 94 13 32 DD 29 05 15  
F6 5E AC B2 14 57 6C 00 FB F3 90 2C 7E 8F 33 09 60 EB EC 99 5C CD 15 71 45 CC CF 2D 43 0A AF 55 84 39 DF AF 78 19 F7 A8 BC B3 30 C6 42 13 BB 66 B6 52 7A 00 F6 85 56 93 A2 DF BD 8D C7 84 26 48 A3 7A 84 49 D9 2A 13 70 C7 DC 03 C3 58 6E  
2D 8A 8B BD F9 05 D4 63 E1 FC 35 82 4F 90 98 5C AB 90 32 E5 A7 B5 58 20 C6 03 9F 26 D4 7A 32 7B 1D DB BC 97 E1 43 E7 3E 21 46 61 79 37 45 F8 90 B6 FB 61 AE 44 A1 D9 62 79 85 26 3D 37 D4 40 BE 29 E1 61 60 94 0C 47 88 C9 21 21 E8 8C  
E4 C8 A4 D5 F3 C8 D8 F9 SA DF C3 0A 9F 27 E4 C6 05 53 F5 83 38 B9 5A  
  
server\_name www.higginbotham.com  
grease [0x3a3a] 00

*Source: Fiddler Capture*

| Headers  | TextView  | SyntaxView   | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second encryption algorithm |
|----------|---|--|----------|---------|------|---------|-----|------|-----|-----------------------------|
| 00000019 | 63 6F 6D 3A 34 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77    | com:443 HTTP/1.1..Host: www.higginbotham.com:443..Connection: keep-alive..User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36... |          |         |      |         |     |      |     |                             |
| 00000032 | 77 77 2E 68 69 67 69 6E 62 6F 74 68 61 6D 2E 63 6F 6D 3A 34 34 33 0D 0A       | .A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below...Secure   |          |         |      |         |     |      |     |                             |
| 0000004B | 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 55    | Protocol: TLS 1.3.Cipher Suite: TLS_AES_256_GCM_SHA384..Record Layer Version: 3.3 (TLS/1.2).Random:  |          |         |      |         |     |      |     |                             |
| 00000064 | 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57    | F6 9D 1E 05 3D 58 53 50  |          |         |      |         |     |      |     |                             |
| 0000007D | 69 6E 64 6F 77 73 20 4E 54 20 31 30 2E 30 3B 20 57 69 6E 36 34 3B 20 78 36    | C5 38 CB 68 E9 B1 71 BE 0  |          |         |      |         |     |      |     |                             |
| 00000096 | 34 29 20 41 70 70 6C 65 57 65 62 4B 69 74 2F 35 33 37 2E 33 36 20 28 4B 48    | 2 77 A7 FA AB 3F CC 1D 97  |          |         |      |         |     |      |     |                             |
| 000000AF | 54 4D 4C 2C 20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72 6F 6D 65 2F 31    | 1B 4C AF AB 7A CD 69."Time": "21-09-1972 08:33:18.   |          |         |      |         |     |      |     |                             |
| 000000C8 | 32 36 2E 30 2E 30 20 53 61 66 61 72 69 2F 35 33 37 2E 33 36 0D 0A 0D          | SessionID: ME B3 4B 70 12  |          |         |      |         |     |      |     |                             |
| 000000E1 | 0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 43 6C 69 65 6E    | 4D 2C CB 6A 5B 9A 63 89  |          |         |      |         |     |      |     |                             |
| 000000FA | 74 48 65 6C 6C 6F 20 68 61 6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E    |  |          |         |      |         |     |      |     |                             |
| 00000113 | 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74 65 64 20 74 68 65 20    |  |          |         |      |         |     |      |     |                             |
| 0000012C | 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 53 65 63 75 72 65    |  |          |         |      |         |     |      |     |                             |
| 00000145 | 20 50 72 6F 74 6F 63 6F 6C 3A 20 54 4C 53 20 31 2E 33 0A 43 69 70 68 65 72    |  |          |         |      |         |     |      |     |                             |
| 0000015E | 20 53 75 69 74 65 3A 20 54 4C 53 5F 41 45 53 5F 32 35 36 5F 47 43 4D 5F 53    |  |          |         |      |         |     |      |     |                             |
| 00000177 | 48 41 33 38 34 0A 0A 52 65 63 6F 72 64 20 4C 61 79 65 72 20 56 65 72 73 69    |  |          |         |      |         |     |      |     |                             |
| 00000190 | 6F 6E 3A 20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 52 61 6E 64 6F 6D 3A    |  |          |         |      |         |     |      |     |                             |
| 000001A9 | 20 46 36 20 39 44 20 31 45 20 30 35 20 33 44 20 35 38 20 35 33 20 35 30 20    |  |          |         |      |         |     |      |     |                             |
| 000001C2 | 43 35 20 33 38 20 43 42 20 36 38 20 45 39 20 42 31 20 37 31 20 42 45 20 30    |  |          |         |      |         |     |      |     |                             |
| 000001DB | 32 20 37 37 20 41 37 20 46 41 20 41 42 20 33 46 20 43 43 20 31 44 20 39 37    |  |          |         |      |         |     |      |     |                             |
| 000001F4 | 20 31 42 20 34 43 20 41 46 20 41 42 20 37 41 20 43 44 20 36 39 0A 22 54 69    |  |          |         |      |         |     |      |     |                             |
| 0000020D | 6D 65 22 3A 20 32 31 2D 30 39 2D 31 39 37 32 20 30 38 3A 33 33 3A 31 38 0A    |  |          |         |      |         |     |      |     |                             |
| 00000226 | 53 65 73 73 69 6F 6E 49 44 3A 20 35 45 20 42 33 20 34 42 20 37 30 20 31 32    |  |          |         |      |         |     |      |     |                             |
| 0000023F | 20 34 44 20 32 43 20 43 42 20 36 35 41 20 35 42 20 39 41 20 36 33 20 38 39 20 |  |          |         |      |         |     |      |     |                             |

|          | <i>Source: Fiddler Capture</i>   |          |            |          |         |      |         |     |      |     |                           |
|----------|--|----------|------------|----------|---------|------|---------|-----|------|-----|---------------------------|
|          | Headers  | TextView | SyntaxView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML | Second bitstream          |
| 000004C9 | 36 37 20 39 38 20 39 46 20 38 42 20 38 38 20 44 39 20 41 30 20 43 36 20 31 |          |            |          |         |      |         |     |      |     | 67 98 9F 8B 88 D9 A0 C6 1 |
| 000004E2 | 35 20 43 43 20 33 37 20 42 42 20 35 31 20 39 31 20 31 44 20 33 30 20 43 45 |          |            |          |         |      |         |     |      |     | 5 CC 37 BB 51 91 1D 30 CE |
| 000004FB | 20 38 44 20 46 36 20 38 45 20 35 46 20 35 34 20 43 32 20 35 41 20 45 33 20 |          |            |          |         |      |         |     |      |     | 8D F6 8E 5F 54 C2 5A E3   |
| 00000514 | 38 39 20 32 43 20 37 42 20 38 39 20 41 41 20 42 42 20 43 35 20 41 46 20 45 |          |            |          |         |      |         |     |      |     | 9 2C 7B 89 AA BB C5 AF E  |
| 0000052D | 39 20 35 36 20 31 30 20 42 38 20 31 35 20 35 37 20 39 34 20 41 31 20 41 31 |          |            |          |         |      |         |     |      |     | 9 56 10 B8 15 57 94 A1 A1 |
| 00000546 | 20 30 43 20 32 41 20 37 33 20 33 45 20 42 34 20 38 35 20 37 41 20 39 35 20 |          |            |          |         |      |         |     |      |     | OC 2A 73 3E B4 85 7A 95   |
| 0000055F | 34 34 20 33 34 20 30 37 20 31 39 20 43 38 20 33 33 20 33 46 20 45 43 20 39 |          |            |          |         |      |         |     |      |     | 44 34 07 19 C8 33 3F EC 9 |
| 00000578 | 43 20 34 35 20 34 36 20 30 42 20 38 34 20 43 39 20 31 43 20 32 32 20 45 31 |          |            |          |         |      |         |     |      |     | C 45 46 0B 84 C9 1C 22 E1 |
| 00000591 | 20 33 43 20 39 46 20 38 37 20 37 31 20 41 33 20 35 42 20 43 33 20 37 32 20 |          |            |          |         |      |         |     |      |     | 3C 9F 87 71 A3 5B C3 72   |
| 000005AA | 42 30 20 45 31 20 43 39 20 44 35 20 43 41 20 39 44 20 43 34 20 36 39 20 42 |          |            |          |         |      |         |     |      |     | B0 E1 C9 D5 CA 9D C4 69 B |
| 000005C3 | 30 20 30 35 20 35 42 20 39 44 20 45 32 20 41 34 20 32 42 20 35 36 20 46 35 |          |            |          |         |      |         |     |      |     | 0 05 5B 9D E2 A4 2B 56 F5 |
| 000005DC | 20 42 46 20 38 32 20 38 34 20 36 37 20 39 43 20 43 36 20 36 35 20 35 46 20 |          |            |          |         |      |         |     |      |     | BF 82 84 67 9C C6 65 5F   |
| 000005F5 | 38 32 20 33 45 20 42 32 20 46 30 20 42 42 20 46 35 20 33 42 20 41 46 20 30 |          |            |          |         |      |         |     |      |     | 82 3E B2 F0 BB F5 3B AF 0 |
| 0000060E | 46 20 38 36 20 31 31 20 41 34 20 31 43 20 30 30 20 35 41 20 34 36 20 36 39 |          |            |          |         |      |         |     |      |     | F 86 11 A4 1C 00 5A 46 69 |
| 00000627 | 20 44 39 20 35 37 20 41 46 20 33 31 20 43 36 20 43 32 20 46 34 20 46 30 20 |          |            |          |         |      |         |     |      |     | D9 57 AF 31 C6 C2 F4 F0   |
| 00000640 | 33 38 20 46 37 20 45 43 20 39 37 20 36 41 20 32 31 20 37 38 20 43 34 20 41 |          |            |          |         |      |         |     |      |     | 38 F7 EC 97 6A 21 78 C4 A |
| 00000659 | 38 20 41 41 20 42 34 20 38 35 20 43 45 20 43 39 20 38 35 20 36 37 20 44 37 |          |            |          |         |      |         |     |      |     | 8 AA B4 85 CE C9 85 67 D7 |
| 00000672 | 20 37 41 20 30 43 20 45 42 20 41 42 20 37 39 20 31 44 20 38 33 20 33 41 20 |          |            |          |         |      |         |     |      |     | 7A 0C EB AB 79 1D 83 3A   |
| 0000068B | 42 32 20 33 42 20 36 30 20 44 36 20 41 33 20 43 32 20 30 31 20 38 37 20 41 |          |            |          |         |      |         |     |      |     | B2 3B 60 D6 A3 C2 01 87 A |
| 000006A4 | 37 20 46 37 20 43 43 20 42 38 20 41 30 20 45 35 20 31 45 20 31 46 20 35 35 |          |            |          |         |      |         |     |      |     | 7 F7 CC B8 A0 E5 1E 1F 55 |
| 000006BD | 20 38 38 20 33 34 20 36 35 20 32 46 20 42 41 20 43 41 20 30 36 20 39 43 20 |          |            |          |         |      |         |     |      |     | 88 34 65 2F BA CA 06 9C   |
| 000006D6 | 33 42 20 31 37 20 39 44 20 41 36 20 31 31 20 42 32 20 33 39 20 39 46 20 33 |          |            |          |         |      |         |     |      |     | 3B 17 9D A6 11 B2 39 9F 3 |
| 000006EF | 34 20 30 30 20 36 43 20 43 42 20 44 30 20 35 33 20 33 45 20 35 37 20 31 36 |          |            |          |         |      |         |     |      |     | 4 00 6C CB D0 53 3E 57 16 |

*Source: Fiddler Capture*

The solution to the problem is that one never signs an actual message. Rather one signs a value derived from that message. A *cryptographic hash function* is a function that computes a *message authentication code* from a message. The message authentication code is of fixed size, typically 160 or 512 bits long. The function is designed so that it is extremely unlikely that two different messages will correspond to the same code. You may have seen references to the commonly used hash functions MD5, SHA-1, and SHA-256. Suppose that  $H$  is a cryptographic hash function. To sign a message  $m$ , party  $A$  computes  $h = D_A(H(m))$  and sends  $E_B(m, h)$  to  $B$ . Party  $B$  now has evidence that  $A$  signed  $m$  because  $E_A(h) = H(m)$ , and  $A$  is the only one who could have generated a value  $h$  with that property.

<https://cs.pomona.edu/~dkauchak/classes/s17/cs52-s17/handouts/encryption.pdf>

The list of supported symmetric encryption algorithms has been pruned of all algorithms that are considered legacy. Those that remain are all Authenticated Encryption with Associated Data (AEAD) algorithms. The cipher suite concept has been changed to separate the authentication and key exchange mechanisms from the record protection algorithm (including secret key length) and a hash to be used with both the key derivation function and handshake message authentication code (MAC).

<https://datatracker.ietf.org/doc/html/rfc8446#section-4>